Approximation schemes for NP-hard geometric optimization problems: A survey

Sanjeev Arora*
Princeton University

NP-hard geometric optimization problems arise in many disciplines. Perhaps the most famous one is the *traveling salesman problem (TSP)*: given n nodes in $\Re^2$ (more generally, in $\Re^d$), find the minimum length path that visits each node exactly once. If distance is computed using the Euclidean norm (distance between nodes (x_1, y_1) and (x_2, y_2) is $((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2}$) then the problem is called *Euclidean TSP*. More generally the distance could be defined using other norms, such as ℓ_p norms for any $p > 1$. All these are subcases of the more general notion of a geometric norm or *Minkowski norm*. We will refer to the version of the problem with a general geometric norm as *geometric TSP*.

Some other NP-hard geometric optimization problems are *Minimum Steiner Tree* ("Given n points, find the smallest network connecting them,"), *k-TSP*("Given n points and a number k, find the shortest salesman tour that visits k points"), *k-MST* ("Given n points and a number k, find the shortest tree that contains k points"), *vehicle routing, degree restricted minimum spanning tree*, etc. Thus if $P \neq NP$, as is widely conjectured, we cannot design polynomial time algorithms to solve these problems optimally. However, we might be able to design *approximation algorithms*: algorithms that compute near-optimal solutions in polynomial time for *every* problem instance. For $\alpha \geq 1$ we say that an algorithm *approximates* the problem within a factor α if it computes, for every instance I, a solution of cost at most $\alpha \cdot \mathrm{OPT}(I)$, where $\mathrm{OPT}(I)$ is the cost of the optimum solution for I. (The preceding definition is for *minimization problems*; for maximization problems $\alpha \leq 1$.) Sometimes we use the shortened name "α-approximation algorithm."

Bern and Eppstein [14] give an excellent survey circa 1995 of approximation algorithms for geometric problems. The current survey will concentrate on developments subsequent to 1995, many of which followed the author's discovery, in 1996, of a *polynomial time approximation scheme* or "PTAS" for geometric TSP in constant dimensions. (By "constant dimensions" we mean that we fix the dimension d and consider asymptotic complexity as we increase n, the number of nodes.) A PTAS is an "ultimate" approximation

*35 Olden St, Princeton NJ 08544. `aroracs.princeton.edu`. Supported by a David and Lucile Packard Fellowship and NSF grant CCR-0098180

algorithm, or rather, a sequence of algorithms: for each $\epsilon > 0$, the sequence contains a polynomial-time algorithm that approximates the problem within a factor $1 + \epsilon$.

Context. Designing approximation algorithms for NP-hard problems is a well-developed science; see the books by Hochbaum (ed.) [34] and Vazirani [63]. The most popular method involves solving a mathematical programming relaxation (either a linear or semidefinite program) and rounding the fractional solution thus obtained to an integer solution. The bound on the approximation ratio is obtained by comparing to the fractional optimum. However, such methods have not led to PTASs.

Interestingly, work from the last decade on probabilistically checkable proofs (see [8] and references therein) suggests a deeper reason why PTASs have been difficult to design: many problems do not have PTAS's if $P \neq NP$. (In other words, there is some fixed $\gamma > 0$ such that computing $(1 + \gamma)$-approximations for the problem is NP-hard.) This is true for metric TSP and metric Steiner tree, the versions of TSP and Steiner tree respectively in which points lie in a metric space (i.e., distances satisfy the triangle inequality). Trevisan has even shown that Euclidean TSP in $O(\log n)$ dimensions has no PTAS if $P \neq NP$. Thus the existence of a PTAS for Euclidean TSP in constant dimensions —one of the topics of the current survey— is quite surprising[1].

However, a few problems were known to have PTASs even in the 1970s, and many more such problems are now known —such as those described in this survey. In most cases, these PTASs follow a similar design methodology. One proves a "Structure Theorem" about the problem in question, demonstrating the existence of a $(1 + \epsilon)$-approximate solution that has an "almost local" quality (see Section 2 for an example). Such a Structure theorem appears implicitly in descriptions of most earlier PTASs, including the ones for Knapsack [35], planar graph problems [45, 46, 11, 33, 40], and most recently, for scheduling to minimize average completion time [2, 39]. These PTASs involve a simple divide-and-conquer approach or dynamic programming to optimize over the set of "almost local" solutions. We note that even in the context of geometric algorithms, divide-and-conquer ideas are quite old, though they had not resulted in PTASs until recently. Specifically, geometric divide and conquer appears Karp's dissection heuristic [38], Smith's $2^{O(\sqrt{n})}$ time exact algorithm for TSP [59], and Blum, Chalasani and Vempala's approximation algorithm for k-MST [19], and Mata and Mitchell's constant-factor approximations for many geometric problems [47].

Surprisingly, the proofs of the structure theorems for geometric problems are elementary and this survey will describe them essentially completely. We also survey a more recent result of Rao and Smith [55] that improves the running time for some problems. We will be concerned only with asymptotics and hence not with practical implementations. The TSP al-

[1] In fact, the discovery of this algorithm stemmed from the author's inability to extend the results of [8] to Euclidean TSP in constant dimensions.

gorithm of this survey, though its asymptotic running time is nearly linear, is not competitive with existing implementations of other TSP heuristics (e.g., [36, 4]). But maybe our algorithms for other geometric problems will be more competitive.

One of the goals of the current survey is to serve as a tutorial for the reader who wishes to apply the techniques described here to other geometric problems.We break the presentation of the PTAS for the TSP into three sections, Sections 2, 3 and 4 with increasingly sophisticated analyses and corresponding improvements in running time: $n^{O(1/\epsilon)}$, $n(\log n)^{O(1/\epsilon)}$, and $n \log n + n \cdot \epsilon^{-c/\epsilon}$. One reason for this three-step presentation is pedagogical: the simpler analyses are easier to teach, and the simplest analysis —giving an $n^{O(1/\epsilon)}$ time algorithm— may even be suitable for an undergraduate course. Another reason for this three-step presentation becomes clearer in Section 5, when we generalize to other geometric problems. The simplest analysis —since it uses very little that is specific to the TSP— is the easiest to generalize.

Section 5 is meant as a tutorial on how to apply our techniques to other geometric problems. We give three illustrative examples —Minimum Steiner Tree, k-median, and the Minimum Latency Problem, and also include a discussion of some geometric problems that seem to resist our techniques. Finally, Section 6 summarizes known results about many geometric optimization problem together with bibliographic references.

Background on geometric approximation For almost every problem discussed in this survey, Bern and Eppstein describe approximation algorithms that approximate the problem within some constant factor. (One of the exceptions is k-median, for which no constant factor approximation was known at the time a PTAS was found [9].) For the TSP, the best previous algorithm was the *Christofides* heuristic [20], which approximates the problem within a factor 1.5 in polynomial time. The decision version of Euclidean TSP ("Does a tour of cost $\leq C$ exist?") is NP-hard [51, 28], but is not known to be in NP because of the use of square roots in computing the edge costs[2]. Specifically, there is no known polynomial-time algorithm that, given integers $a_1, a_2, \ldots, a_n, C$, can decide if $\sum_i \sqrt{a_i} \leq C$.)

Arora's paper gave the first PTASs for many of these problems in 1996. A few months later Mitchell independently discovered a similar $n^{O(1/\epsilon)}$ time approximation scheme [50]; this algorithm used ideas from the earlier paper of Mata and Mitchell [47]. The running time of Arora's and Mitchell's algorithms was $n^{O(1/\epsilon)}$, but Arora later improved the running time of his algorithm to $n(\log n)^{O(1/\epsilon)}$. Mitchell's algorithm seems to work only in the plane whereas Arora's PTAS works for any constant number of dimensions. If dimension d is not constant but allowed to depend on n, the number of nodes,

[2]For similar reasons, there is no known polynomial time Turing machine algorithm even for Euclidean minimum spanning tree. Most papers in computational geometry skirt this issue by using the Real RAM model.

then the algorithm takes superpolynomial time that grows to exponential around $d = O(\log n)$. This dependence on dimension seems consistent with known complexity results. Trevisan [60] has shown that the Euclidean TSP problem becomes MAX-SNP-hard in $O(\log n)$ dimensions, which means —by the results of [53, 8]— that there is a $\gamma > 1$ such that approximation within a factor $1 + \gamma$ is NP-hard.

1 Introduction to the TSP algorithm

In this section we describe a PTAS for TSP in $\Re^2$ with ℓ_2 norm; the generalization to other norms is straightforward. The algorithm uses divide-and-conquer, and the "divide" part is just the classical quadtree, which partitions the instance using squares that progressively get smaller. Thus the algorithm is reminiscent of Karp's dissection heuristic [38] and a more recent algorithm for TSP in planar graphs due to Grigni et al. [33], which uses a partition theorem about planar graphs. However, the algorithm differs from Karp's dissection heuristic in two ways. First, it uses randomness in constructing the partitions. Second, unlike the dissection heuristic, which treats the smaller squares as independent problem instances (to be solved separately and then linked together in a trivial way) this algorithm allows limited back-and-forth trips between the squares. Thus the subproblems inside adjacent squares are weakly interdependent. Specifically, the algorithm allows $O(1/\epsilon)$ entries/exits to each square of the dissection. A simple dynamic programming finds the best tour with this property. To show that the cost of this tour is within a factor $1 + \epsilon$ of the optimum, we will describe how to transform an optimum tour so that it satisfies the "limited back-and-forth trips" property.

Although our algorithm is described as randomized, it can be derandomized with some loss in efficiency —specifically, by trying all choices for the shifts used in the randomized dissection. Better derandomizations appear in Czumaj and Lingas [21] and Rao and Smith's paper (see Section 3).

1.1 The perturbation

First we perform a simple perturbation of the instance that, without greatly affecting the cost of the optimum tour, ensures that each node lies on the unit grid (i.e., has integer coordinates) and every internode distance is at least 2. Call the smallest axis-parallel square containing the nodes the *bounding box.* Our perturbation will ensure its sidelength is at most $n^2/2$.

We assume $\epsilon > 1/n^{1/3}$; a reasonable assumption since ϵ is a fixed constant independent of the input size. Let d be the maximum distance between any two nodes in the input. Then $2d \leq \text{OPT} \leq nd$. We lay down a grid in which the grid lines are separated by distance $\epsilon d/n^{1.5}$. Then we move each node to its nearest gridpoint. This may merge some nodes; we treat this merged node as a single node in the algorithm. (At the end we will extend

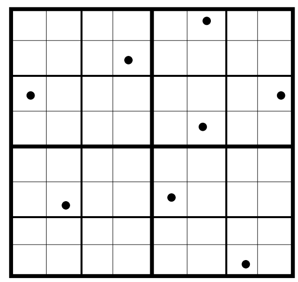

Figure 1: The dissection

the tour to these nodes in a trivial way by making excursions from the single representative.) Note that the perturbation moves each node by at most $\epsilon d/n^{1.5}$, so it affects the optimum tour cost by at most $\epsilon d/n^{0.5}$, which is negligible compared to ϵOPT when n is large. Finally, we rescale distances so that the minimum internode distance is at least 2. Then the maximum internode distance is at most $n^{1.5}/\epsilon$, which is asymptotically less than $n^2/2$, as desired.

1.2 Randomized Dissection

A *dissection* of a square is a recursive partitioning into squares; see Figure 1. We view this partitioning as a tree of squares whose root is the square we started with. Each square in the tree is partitioned into four equal squares, which are its children. The leaves are squares of sidelength 1.

Now we define a *randomized dissection* of the instance. Let $P \in \Re^2$ be the lower left endpoint of the bounding box and let each side have length l. We enclose the bounding box inside a larger square —called the *enclosing box*—of sidelength $L = 2l$ and position the enclosing box such that P has distance a from the left edge and b from the lower edge, where integers $a, b \leq l$ are chosen randomly. We refer to a, b as the *horizontal* and *vertical shift* respectively; see Figure 2. The randomized dissection is the dissection of this enclosing box. Note that we are thinking of the input nodes and the unit grid as being fixed; the randomness is used only to determine the placement of the enclosing box (and its accompanying dissection).

Assume without loss of generality that L is a power of 2 so the squares in the dissection have integer endpoints and leaf squares have sides of length 1 (and hence at most one node in them). The *level* of a square in the dissection is its depth from the root; the root square has level 0. We also assign a *level* from 0 to $\log L - 1$ to each horizontal and vertical grid line that participated in the dissection. The horizontal (resp., vertical) line that divides

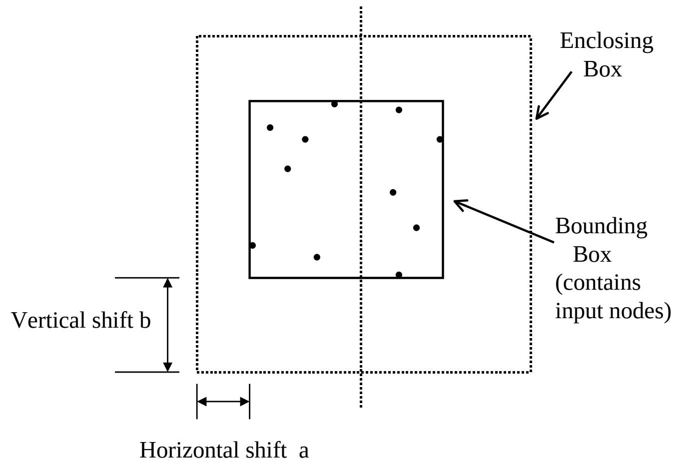

Figure 2: The enclosing box contains the bounding box and is twice as large, but is shifted by random amounts in the x and y directions.

the enclosing box into two has level 0. Similarly, the 2^i horizontal and 2^i vertical lines that divide the level i squares into level $i+1$ squares each have level i.

The following property of a random dissection will be crucial in the proof (see for example the proof of Lemma 2). Consider any fixed vertical grid line that intersects the bounding box of the instance. What is the chance that it becomes a level i line in the randomized dissection? There are 2^i values of the horizontal shift a (see Figure 2 again) that cause this to happen, so

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

Thus the randomized dissection treats —in an expected sense— all such grid lines symmetrically.

1.3 Portal-respecting tours

Each grid line will have special points on it called *portals*. A level i line has $2^{i+1}m$ equally spaced portals, where m is the *portal parameter* (to be specified later). We require m to be a power of 2. In addition, we also refer to the corners of each a square as a portal. Since the level i line has 2^{i+1} level $i+1$ squares touching it, we conclude that each side of the square has at most $m+2$ portals (m usual portals, and the 2 corners), and a total of at most $4m+4$ portals on its boundary. A *portal-respecting* tour is one that, whenever it crosses a grid line, does so at a portal. Of course, such a tour will not be optimal in general, since it may have to deviate from the straight-line path between nodes.

A portal-respecting tour is *k-light* if it crosses each side of each dissection square at most k times. The optimum portal-respecting tour does not need to visit any portal more than twice; this follows by the standard observation that removing repeated visits can, thanks to triangle inequality, never increase the cost. Thus the optimum portal-respecting tour is $(m+2)$-light. A simple dynamic programming can find the optimum portal-respecting tour in time $2^{O(m)} L \log L$. Allowing $m = O(\log n/\epsilon)$ suffices (see Section 2), hence the running time is $n^{O(1/\epsilon)}$.

A more careful analysis in Section 3 shows that actually we only need to consider portal-respecting tours that are k-light where $k = O(1/\epsilon)$. Our dynamic programming can find the best such tour in poly$(\binom{m}{k})2^{O(k)} L \log L$ time, which is $O(n(\log n)^{O(1/\epsilon)})$.

Dynamic programming. We sketch the simple dynamic programming referred to above. To allow a cleaner analysis later, the randomized dissection was described above as a regular 4-ary tree in which each leaf has the same depth. In an actual implementation, however, one can truncate this tree so that the partitioning stops as soon as a square has at most 1 input node in it. Then the dissection has at most $2n$ leaves and hence $O(n \log n)$ squares. Furthermore, the cost of the optimum portal respecting tour cannot increase since truncating the dissection can only reduce the number of times the tour has to make detours to pass through portals.

The truncated dissection (which is just the quadtree of the enclosing box) can be efficiently computed, for instance by sorting the nodes by x- and y-coordinates (for better algorithms, especially in higher dimensions, see Bern et al. [15]). The dynamic programming now is the obvious one. Suppose we are interested in portal-respecting tours that enter/exit each dissection square at most $4k$ times. The subproblem inside the square can be solved independently of the subproblem outside the square so long as we know the portals used by the tour to enter/exit the square, and the *order* in which the tour uses these portals. Note that given this *interface* information, the subproblems inside and outside the square involve finding not salesman tours but a set of up to $4k$ vertex-disjoint paths that visit all the nodes and visit portals in a way consistent with the interface. (See Figure 3.) We maintain a lookup table that, for each square and for each choice of the interface, stores the optimum way to solve the subproblem inside the square. The lookup table is filled up in a bottom-up fashion in the obvious way. Clearly, its size is (# of dissection squares) $\times\, m^{O(k)} k!$.

One can actually reduce the $m^{O(k)} k!$ term to $2^{O(m)} = n^{O(1/\epsilon)}$ by noticing that the dynamic programming need not consider all possible interfaces for a square since the optimum portal respecting tour in the plane does not cross itself. Interfaces corresponding to tours that do not cross themselves are related to well-matched parenthesis pairs, and the number of possibili-

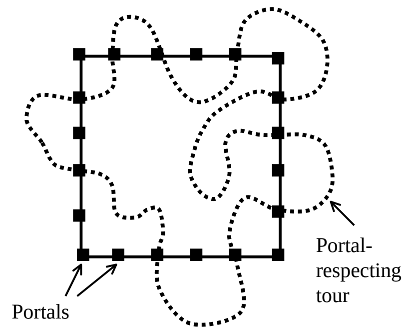

Figure 3: This portal-respecting tour enters and leaves the square 10 times, and the portion inside the square is a union of 5 disjoint paths.

ties for these are given by the well-known *Catalan* numbers. We omit details since the better analysis given in Section 3 reduces k to $O(1/\epsilon)$, which makes m^k much smaller than $2^{O(m)}$.

2 Structure Theorem: First Cut

First we give a very simple analysis (essentially from [9]) showing that if the portal parameter m is $O(\log n/\epsilon)$, then the best portal-respecting tour is likely to be near optimal. This tour may enter/leave each square $8m + 8$ times.

Let OPT denote the cost of the optimum salesman tour and $\text{OPT}_{a,b,m}$ denote the cost of the best portal-respecting tour when the portal parameter is m and the random shifts are a, b. Our notation stresses the dependence of this number upon shifts a and b and the portal parameter m. Clearly, $\text{OPT}_{a,b,m} \geq \text{OPT}$.

Theorem 1 *The expectation (over the choice of a, b) of $OPT_{a,b,m} - OPT$ is at most $2 \log L/m OPT$, where L is the sidelength of the enclosing box.*

Consequently, the probability is at least $1/2$ (over the choice of shifts a, b) that the difference $\text{OPT}_{a,b,m} - \text{OPT}$ is at most twice its expectation, namely $4 \log L/m \cdot \text{OPT}$. When the root square has sides of length $L \leq n^2$ (as ensured by our perturbation) and m is at least $8 \log n/\epsilon$, this difference is at most $8 \log n/m \cdot \text{OPT} = \epsilon \cdot \text{OPT}$. Thus $\text{OPT}_{a,b,m} \leq (1 + \epsilon)\text{OPT}$ with probability at least $1/2$.

The following simple lemma lies at the heart of Theorem 1. It analyses the expected length increase when a single edge it is made portal-respecting. Theorem 1 immediately follows by linearity of expectations, since the tour length is a sum of edge lengths.

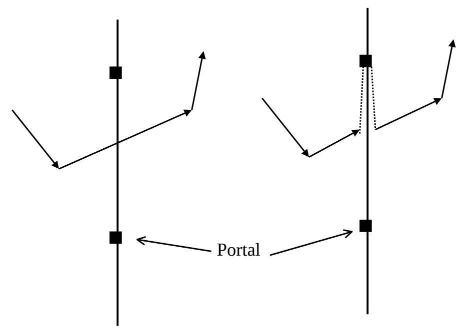

Figure 4: Every crossing is moved to the nearest portal by adding a "detour."

For any two nodes $u, v \in \Re^2$ let the *portal-respecting distance* between u and v, denoted $d_{a,b,m}(u,v)$ be the shortest distance between them when all intermediate grid lines have to be crossed at portals.

Lemma 2 *When shifts a, b are random, the expectation of $d_{a,b,m}(u,v) - d(u,v)$ is at most $\frac{2\log L}{m} d(u,v)$, where L is the sidelength of the enclosing box.*

Proof: The expectation, though difficult to calculate exactly, is easy to upperbound. The straight line path from u to v crosses the unit grid at most $2d(u,v)$ times. To get a portal-respecting path, we move each crossing to the nearest portal on that grid line (see Figure 4), which involves a detour whose length is at most the interportal distance. If the line in question has level i, the interportal distance is $L/m2^{i+1}$. By (1), the probability that the line is at level i is $2^{i+1}/L$. Hence the expected length of the detour is at most

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

The same upperbound applies to each of the $2d(u,v)$ crossings, so linearity of expectations implies that the expected increase in moving all crossings to portals is at most $2d(u,v)\log L/m$. This proves the lemma. $\Box$

3 Structure Theorem: Second Cut

Recall that a portal-respecting tour is *k-light* if it crosses each side of each dissection square at most k times. Let $\mathrm{OPT}_{a,b,k,m}$ denote the cost of the best such salesman tour when the portal paramter is m.

Theorem 3 *$E[OPT_{a,b,k,m} - OPT] \le (\frac{2\log L}{m} + \frac{12}{k-5})OPT$, where the expectation is over the choice of shifts a, b.*

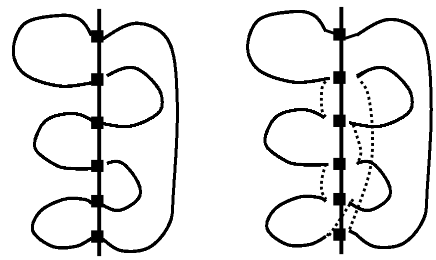

Figure 5: The tour crossed this line segment 6 times, but breaking it and reconnecting on each side (also called “patching”) reduced the number of crossings to 2.

Thus if $m = \Omega(\log n/\epsilon)$ and $k \geq 24/\epsilon + 5$, the probability is at least $1/2$ that the best k-light tour has cost at most $(1+\epsilon)$OPT.

The analysis in this proof has a global nature, by which we mean that it takes all the edges of the tour into account simultaneously (see our charging argument below). By contrast, many past analyses of approximation algorithms for geometric problems — see for instance algorithms surveyed in [14] and also our simpler analysis in Section 2— reason in an edge-by-edge fashion.

We will use the following well-known fact about Euclidean TSP that is implicit in the analysis of Karp’s dissection heuristic (or even [13]), and is made explicit in [5].

Lemma 4 (Patching Lemma) *Let S be any line segment of length s and π be a closed path that crosses S at least thrice. Then we can break the path in all but two of these places, and add to it line segments lying on S of total length at most $3s$ such that π changes into a closed path π' that crosses S at most twice.*

Proof: For simplicity we give a proof using segments of length $6s$ instead of $3s$; the proof of $3s$ uses the Christofides heuristic and the reader may wish to work it out.

Suppose π crosses S a total of t times. Let $M_1, \ldots, M_t$ be the points on which π crosses S. Break π 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 $2j$ 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 $M_1, \ldots, M_{2j}$. 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 π. We think of J' as lying on the left of S and J'' as lying on the right of S.

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

Together with the paths $P_1, \ldots, P_{2j}$, 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. (See Figure 5.) Hence we have proved the theorem. $\square$

Another needed fact —implicit also in Lemma 2—relates the cost of a tour to the total number of times it crosses the lines in the unit grid. If l is one of these lines and π is a salesman tour then let $t(\pi, l)$ denote the number of times π crosses l. Then

$$\sum_{l:\text{vertical}} t(\pi, l) + \sum_{l:\text{horizontal}} t(\pi, l) \leq 2\,\text{cost}(\pi) \tag{2}$$

Now we are ready to prove the main result of the section.

Proof:(Theorem 3) The main idea is to transform an optimum tour π into a k-light tour. Whenever the tour enters/exits a square "too many" times, we use the Patching Lemma to reduce the number of crossings. This increases cost, which we upperbound in the expectation by using the relationship in (2).

Let us describe this tour transformation process for a single vertical grid line, say l. (A similar transformation happens at every grid line.) Suppose l has level i. It is touched by 2^{i+1} level $i + 1$ squares, which partition it into 2^{i+1} segments of length $L/2^{i+1}$. For each $j > i$, line l is also touched by 2^j level j squares. In general, we will refer to the portion of l that lies in a level j square as a *level j segment.* The final goal is to reduce the number of crossings in each level i segment to k or less; we do this as follows.

Let $s = k - 4$. An *overloaded* segment of l is one which the tour crosses at least $s + 1$ times. The tour transformation proceeds as follows. For every level $\log L - 1$ segment that is overloaded, we apply the patching lemma and reduce the number of crossings to 2. In the transformed tour, we now look at segments of level $\log L - 2$ and for each overloaded segment, apply the patching lemma to reduce the number of crossings to 2. Continuing this way for progressively higher levels, we stop when no segment at level i is overloaded. At the end, we move all crossings to portals; we do this by adding vertical detours (i.e., vertical segments), as in Figure 4.

To analyse the cost increase in this transformation, we consider an imaginary procedure in which the tour transformation on this vertical grid line l proceeds to level 0, i.e., until the entire line is not overloaded. Let $X_{l,j}(b)$ be a random variable denoting the number of overloaded level j segments encountered in this imaginary procedure. We draw attention to the fact that

$X_{l,j}(b)$ is determined by the vertical shift b alone, which determines the location of the tour crossings with respect to the segments on l. We claim that for every b,

$$\sum_{j\geq 0} X_{l,j}(b) \leq \frac{t(\pi,l)}{s-1}. \tag{3}$$

The reason is that the optimum tour π crossed grid line l only $t(\pi,l)$ times, and each application of the Patching Lemma counted on the left hand side of (3) replaces at least $s+1$ crossings by at most 2, thus eliminating $s-1$ crossings each time.

Since a level j segment has length $L/2^j$, the cost of the imaginary transformation procedure is, by the Patching Lemma, at most

$$\sum_{j\geq 1} X_{l,j}(b) \cdot \frac{3L}{2^j}. \tag{4}$$

(Note that in (4) we omit the case $j = 0$ because the level 0 square is just the bounding box, and the tour lies entirely inside it.)

The *actual* cost increase in the tour transformation at l depends on the level of l, which is determined by the horizontal shift a. When the level is i, the terms of (4) corresponding to $j \geq i+1$ upperbound the cost increase:

$$\text{Increase in tour cost when } l \text{ has level } i \leq \sum_{j\geq i+1} X_{l,j}(b) \cdot \frac{3L}{2^j}, \tag{5}$$

We "charge" this cost to l. Of course, whether or not this charge occurs depends on whether or not i is the level of line l, which by (1) happens with probability at most $2^{i+1}/L$ (over the choice of the horizontal shift a). Let Y_a be a random variable

$$Y_{l,a} = \text{charge to } l \text{ when horizontal shift is } a.$$

Thus for every vertical shift b

$$\begin{aligned}
E_a[Y_{l,a}] &= \sum_{i\geq 1} \frac{2^{i+1}}{L} \cdot (\text{charge to } l \text{ when its level is } i) \\
&\leq \sum_{i\geq 1} \frac{2^{i+1}}{L} \cdot \sum_{j\geq i+1} X_{l,j}(b) \cdot \frac{3L}{2^j} \\
&= 3 \cdot \sum_{j\geq 1} \frac{X_{l,j}(b)}{2^j} \cdot \sum_{i\leq j-1} 2^i \\
&= 3 \cdot \sum_{j\geq 1} \frac{X_{l,j}(b)}{2^j} \cdot (2^j - 1) \\
&\leq 3 \cdot \sum_{j\geq 1} 2 \cdot X_{l,j}(b) \\
&\leq \frac{6\, t(\pi,l)}{s-1}
\end{aligned}$$

We may now appear to be done, since linearity of expectations seems to imply that the total expected cost charged to all lines is

$$E_a[\sum_l Y_{l,a}] = \sum_l \frac{6t(\pi, l)}{s-1}, \tag{6}$$

which from (2) is at most

$$\leq 12\frac{\text{cost}(\pi)}{s-1}. \tag{7}$$

However, we are not done. We have to worry about the issue of how the modifications at various grid lines affect each other. Fixing overloaded segments on vertical grid lines involves adding vertical segments to the tour, thus increasing the number of times the tour crosses horizontal grid lines; see Figure 5. Then we fix the overloaded segments on horizontal grid lines. This adds horizontal segments to the tour, which may potentially lead to some vertical lines becoming overloaded again. We need to argue that the process stops. To this end we make a simple observation: fixing the overloaded segments of the vertical grid line l adds at most 2 additional crossings on any horizontal line l'. The reason is that if the increase were more than 2, we could just use the Patching Lemma to reduce it to 2 and this would not increase cost since the Patching Lemma is being invoked for segments lying on l which have zero horizontal separation (that is, they lie on top of each other). Also, to make sure that we do not introduce new crossings on l itself we apply the patching separately on vertical segments of both sides of l. Arguing similarly about all intersecting pairs of grid lines, we can ensure that at the end of all our modifications, the tour crosses each side of each dissection square up to $s+4$ times; up to s times through the side and up to 4 times through the two corners. Since $s+4=k$, we have shown that the tour is k-light.

The analysis of the cost incurred in moving all crossings to the nearest portal is similar to the one in Section 2 and gives the $\frac{2\log L}{m}$OPT term in the statement of Theorem 3. This completes our proof.

□

4 Rao-Smith Algorithm

Rao and Smith [55] describe an improvement of the above algorithm that runs in time $O(n\log n + n2^{\text{poly}(1/\epsilon)})$. First they point out why the running time of the above algorithm is $n(\log n)^{O(1/\epsilon)}$. It is actually $nm^{O(k)}$ where m is the portal parameter and k is the number of times the tour can cross each side of each dissection square. The n comes from the number of squares in the dissection, and $m^{O(k)}$ from the fact that the dynamic programming has to enumerate all possible "interfaces," which involves enumerating all ways of choosing k crossing points from among the $O(m)$ portals. The analysis given above seems to require m to be $\Omega(\log n)$ and k to be $\Omega(1/\epsilon)$, which makes the running time $\Omega(n(\log n)^{O(1/\epsilon)})$. Their main new idea is to reduce the $m^{O(k)}$ enumeration time by giving the dynamic programming more hints about which portals are used by the tour to enter/exit each dissection square.

The "hints" mentioned above are generated using a $(1+\epsilon)$-*spanner* of the input nodes. This is a connected graph with $O(n/\text{poly}(\epsilon))$ edges in which the distance between any pair of nodes is within a factor $(1+\epsilon)$ of the Euclidean distance. Such a spanner can be computed in $O(n \log n/\text{poly}(\epsilon))$ time (see Althoefer et al. [3]). Note that distances in the spanner define a metric space in which the optimum TSP cost is within a factor $(1+\epsilon)$ of the optimum in the Euclidean space.

Rao and Smith notice that the tour transformation procedure of Section 3 can be applied to any connected graph, and in particular, to the spanner. The transformed graph is portal-respecting, as well as k-light for $k = O(1/\epsilon)$. The *expected* distance between any pair of points in the transformed graph is at most factor $(1+\epsilon)$ more than what it was in the spanner. Note that some choices of the random shifts may stretch the distance by a much larger factor; the claim is only about the expectation. (In particular, it is quite possible that the transformed graph is not a spanner.) By linearity of expectation, the expected increase in the cost of the optimum tour in the spanner is also at most a factor $(1+\epsilon)$.

Now the algorithm tries to find the optimum salesman tour with respect to distances in this transformed graph. Since the transformed graph is k-light, we know for each dissection square a set of at most $4k$ portals that are used by the tour to enter/exit the square. As usual, we can argue that no portal is crossed more than twice. Thus the dynamic programming only needs to consider $k^{O(k)}$ "interfaces" for each dissection square. For more details see Rao and Smith's paper.

4.1 Higher-dimensional versions

Although we concentrated on the version of the Euclidean TSP in $\Re^2$, the algorithms also generalize to higher-dimensional versions of all these problems. The analysis is similar, with obvious changes such as replacing squares by higher dimensional cubes. For more details see Arora [5].

Note that the running time —even after using the ideas of Rao and Smith— has a doubly exponential dependence upon the dimension d. So the dimension should be $o(\log \log n)$ in order for the running time to be polynomial. We have reason to believe that this dependence is inherent, since Trevisan [60] has shown that there is an $\epsilon > 0$ such that $(1+\epsilon)$-approximation to Euclidean TSP in n dimensions is MAX-SNP hard. Using a general dimension-reduction due to Johnson and Lindenstrauss [37], it follows that if a polynomial-time approximation scheme exists even in

$O(\log n)$ dimensions, then P = NP.

5 Generalizing to other problems: a methodology

The design of the PTAS for the TSP uses very few properties specific to the TSP. Below, we abstract out these properties, and identify other problems that share these properties. The first property of the TSP that we needed was that the objective is a sum of edge lengths.

Next, we crucially needed the fact that the notion of portal-respecting solutions (whereby edges have to cross the boundaries of quadtree squares only at portals) makes sense. We used this in Section 2 by showing that every set of edges in the plane can be made portal-respecting without greatly affecting their total length. We abstract out this statement in the following Theorem, which follows from Lemma 2 by linearity of expectations.

Theorem 5 *Let E be any set of edges in the plane in which each edge has length at least 4 and all edges lie inside a square with a side of length L. When we pick shifts $a, b \in \{0, 1, , \dots, L-1\}$ randomly then the edges can be made portal-respecting to the shifted quadtree with expected cost increase $\frac{2\log L}{m} cost(E)$. Here m is the portal parameter and L is the sidelength of the enclosing box.*

The more sophicated analysis of Section 3 relied on another property of the TSP, embodied in the Patching Lemma (Lemma 4). This property was needed in the proof to transform the optimum tour (over many steps and without greatly affecting the cost) into a k-light tour where k is $O(1/\epsilon)$. Thus our dynamic programming can restrict attention to k-light tours, and thus be more efficient. The Patching Lemma also holds for holds for trees and Steiner trees. It may also hold (possibly in a weaker form) for many other geometric problems involving routing or connecting.

General methodology. The discussion above suggests the following general methodology for finding out if a geometric problem may have an approximation scheme.

1. Check if the objective function involves a sum of edge lengths.
2. Check if the notion of portal-respecting solutions makes sense. (That is, can one turn a portal-respecting solution into a valid solution?)
3. Check if one can describe a small "interface" between adjacent squares that allows the subproblems inside them to be solved independently. If so, one can probably use dynamic programming to get an approximation scheme that runs in $n^{O(\log n/\epsilon)}$ time or better.

4. If the above properties hold, check if the Patching Lemma (or something akin to it) holds. If so, the proof of Section 3 can probably be made to work for the problem and also the proof of Section 4

Now we illustrate how to apply this methodology using three geometric problems. Our examples were chosen carefully. Minimum Steiner Tree (Section 5.1) satisfies all the properties mentioned above so the TSP algorithm generalizes to it in a straightforward way. In the k-median problem (Section 5.2) the objective function is a sum of edge lengths but the Patching Lemma does not hold for this problem. However, the notion of portal-respecting solution makes sense for it, and we obtain a PTAS. Finally, the Minimum Latency problem (Section 5.3) is interesting because it neither obeys the Patching Lemma, nor is its objective function a sum of edge lengths. Nevertheless, by a deeper analysis of the objective function, we can write it as a weighted sum of salesmen paths, and then restrict attention to portal-respecting solutions.

Finally, in Section 6.4 we describe two interesting problems for which the methodology has not yet worked.

5.1 Minimum Steiner Tree

In the *Minimum Steiner Tree* problem, we are given n nodes in $\Re^d$ and desire the minimum-cost tree connecting them[3]. In general, the minimum spanning tree is not an optimal solution and one needs to introduce new points (called "Steiner" points) as nodes in the solution. In case of three nodes at the corners of an equilateral triangle in $\Re^2$ (with distances measured in ℓ_2 norm), the optimum Steiner tree contains the centroid of the triangle, and has cost $\sqrt{3}/2$ factor lower than the MST. Furthermore, the famous Gilbert-Pollak [29] conjecture said that for every set of input nodes, a Steiner tree has cost at least $\sqrt{3}/2$ times the cost of the MST. Du and Hwang [23] proved this conjecture and thus showed that the MST is a $2/\sqrt{3}$-approximation to the optimum Steiner tree. A spate of research activity in recent years starting with the work of Zelikovsky[65] has provided better approximation algorithms, with an approximation ratio around 1.143 [66]. The metric case does not have an approximation scheme if $P \neq NP$ [16].

The Steiner Tree problem involves an objective function that is a sum of edge lengths and it obeys the Patching Lemma (as is easily checked). Now we briefly describe the algorithm.

First we perturb the instance to ensure that all coordinates are integers and the ratio of the maximum internode distance to the smallest internode distance is $O(n^2)$. We proceed exactly as for the TSP. If d denotes the maximum internode distance, lay a grid of granularity $\epsilon d/n^{1.5}$ and move every node to its nearest grid point. We also restrict Steiner nodes to lie on grid points. As is well-known, the optimum Steiner tree has at most $n - 1$

[3]It appears that this problem was first posed by Gauss in a letter to Schumacher [32].

Steiner nodes (and hence a total of at most $2n - 1$ edges), so the cost of the optimum solution changes by at most $(2n - 1)\epsilon d/n^{1.5}$, which is less than $\epsilon \cdot OPT/2$ as n grows.

We define a randomized dissection as well as portals in the same way as we did for the TSP. A k-light portal-respecting Steiner tree is one that crosses grid lines only at portals and which enters and leaves each side of each square in the dissection at most k times. (Note that portals have a natural meaning for the Steiner tree problem: they are Steiner nodes!) We can find the best such tree by dynamic programming similar to the one for the TSP. There are only two modifications. First, the base case of the dynamic programming, involving the smallest squares in the dissection, has to consider the possibility of using Steiner nodes in the optimum solution. For this it needs to run an exponential-time algorithm for the Steiner forest problem. Luckily, the Steiner forest problem inside this square has constant size, specifically, at most $4k + 1$ (at most $4k$ portals and at most 1 input node). The other modification to the dynamic program is in the way of specifying the "interface" between adjacent squares of the dissection, since the final object being computed is a tree and not a tour. The details are straightforward and left to the reader.

The proof of correctness is essentially unchanged from the TSP case since the Patching Lemma holds for the Steiner Tree problem.

5.2 k-median

In the k-median problem we are given n nodes $\{x_1, x_2, \ldots, x_n\}$ and a positive integer k, and we have to find a set of k medians $\{m_1, m_2, \ldots, m_k\}$ that minimizes

$$\sum_{i=1}^{n} \min_{1 \le j \le k} \left\{ d(x_i, m_j) \right\} \tag{8}$$

where $d(\cdot, \cdot)$ denotes distance. By grouping together terms in (8) corresponding to x_i's which have the same nearest median (breaking ties arbitrarily) we see that the problem involves putting k "stars" (i.e., graphs in which nodes are attached to a single center) in $\Re^2$ which cover all n input nodes; the medians are at the centers of these stars. Thus we may think of the k-median problem as *covering by k stars*. There are two variations of the problem depending upon whether or not the medians are required to be input nodes. The algorithm we are going to describe works for both variations.

The Patching Lemma does not hold for the k-median problem: given a the optimum solution —a union of k stars— and a straight line segment in the plane, there is no general way to reduce the number of star edges crossing the straight line without raising the cost by a lot.

However, the notion of a portal-respecting solution — one in which the edges of the stars, whenever they cross the edges of the quadtree, do so at a

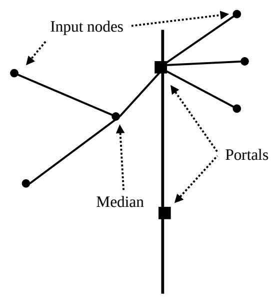

Figure 6: A portal-respecting solution for k-median.

portal— makes sense, as we see below. Using Theorem 5 We can show that there is a portal-respecting solution of cost at most $(1+\epsilon)OPT$.

A simple perturbation allows us to assume that the bounding box has length $O(n^4)$ (see [9]) and all nodes and medians have integer coordinates. Then by making the portal parameter $m = \Omega(\log n/\epsilon)$, the expected cost of the optimum portal-respecting cover by k stars is at most $(1+\epsilon)$OPT. Now we describe the dynamic programming to find the optimum portal-respecting cover by k stars. As usual, for each square, we have to decide upon an "interface" between the solutions inside and outside the square, so that the DP can solve the subproblem inside the square independently of the one outside. Since all star edges have to enter or leave the square via a portal, the algorithm can solve the subproblem inside the square so long as it has been told (a) the number of medians that lie inside the square, and (b) the distance from each portal to the nearest median outside the square. This is enough information to solve the subproblem inside the square optimally.

Specifying (a) requires $\lceil \log k \rceil$ bits and specifying (b) requires $O(\log^2 n/\epsilon)$ bits because each median has integer coordinates. So there are $2^{O(\log^2 n/\epsilon)}$ possible interfaces. Since the number of squares in the dissection is $O(n \log n)$, the running time is essentially the same as the number of possible interfaces. Thus we get an approximation scheme that runs in $n^{O(\log n/\epsilon)}$ time, which is slightly superpolynomial. By a more careful choice for the interface and other tricks, the running time can be made polynomial [9] and even near linear [42].

5.3 Minimum Latency

The *minimum latency problem*, also known as *traveling repairman problem* [1], is a variant of the TSP in which the starting node of the tour is given and the goal is to minimize the sum of the *arrival times* at the other nodes. (The arrival time is the distance covered before reaching that node.)

Like the TSP, the problem is NP-hard in the plane. But it has a reputation for being much more difficult than the TSP: the class of tractable instances consists only of path graphs [1]. The problem on weighted trees is NP-hard. By contrast, the TSP can be optimally solved on a tree. The metric case of the latency problem is MAX-SNP-hard (this follows from the reduction that proves the MAX-SNP-hardness of TSP(1, 2) [53]), and therefore the results of Arora et al. [8] imply that unless P = NP, a PTAS does not exist for metric instances. Blum, Chalasani, Coppersmith, Pulleyblank, Raghavan and Sudan gave a 144-approximation algorithm for the metric case and a 8-approximation for weighted trees. Goemans and Kleinberg [31] then gave a 21.55-approximation in the metric case and a 3.59..-approximation in the geometric case (the latter uses the PTAS for k-TSP). Arora and Karakostas [7] designed a quasipolynomial-time approximation scheme for the problem. We do not know whether the running time can be reduced to polynomial.

Consider the objective function for the problem: we have to find a permutation π of the n nodes such that $\pi(1) = 1$ and we minimize

$$\sum_{i=2}^{n} \sum_{j=1}^{i-1} d(\pi(j), \pi(j+1)).$$

Note the nonlocal nature of the objective function: an extra edge inserted at the beginning of the tour affects the latency of *all* the remaining nodes. For this reason, the shortest salesman tour may be very suboptimal in terms of its latency, as the reader may wish to verify. This strange objective function also poses problems in applying our general methodology, since the it is not a simple sum of edge lengths! However, we show below that the objective may be approximated as a weighted sum of $O(\log n/\epsilon)$ salesman paths, and then we can use all our usual techniques by recoursing to linearity of expectations.

Now we describe this argument of Arora and Karakostas. Let $\epsilon > 0$ be any parameter such that we desire a $(1+\epsilon)$-approximation. First, by a simple perturbation —merge together pairs of nodes separated by distance at most $O(\epsilon L/n^2)$, where L is the largest internode distance— we may assume that the minimum nonzero internode distance is 4 and maximum internode distance is $O(n^2/\epsilon)$. Since ϵ is constant, we will often think of the maximum internode distance as $O(n^2)$.

We will show that to find a $(1+\epsilon)$-approximate minimum latency tour, it suffices to find the tour as a union of $O(\log n/\epsilon)$ segments, where the ith segment contains n_i nodes, and the numbers $n_1, n_2, \ldots$ decrease geometrically and depend on n, ϵ only (and not on the input nodes). Within each segment the order of visits to the nodes does not matter, as long as the total length is close to minimum.

Let $\mathcal{T}$ be an optimal tour with total latency OPT. Imagine breaking this tour into r segments, so that in segment i we visit n_i nodes, where

$$\begin{aligned}
n_i &= \left\lceil (1+\epsilon)^{r-1-i} \right\rceil \text{ for } i = 1 \ldots r-1 \\
n_r &= \lceil 1/\epsilon \rceil .
\end{aligned}$$

Let the length of the ith segment be T_i. If we let $n_{>i}$ denote the total number of nodes visited in segments numbered $i+1$ and later, then a simple calculation shows that (and this was the reason for our choice of n_i's)

$$n_{>i} = \sum_{j>i} n_j \le \frac{n_i}{\epsilon}, \text{ for every } i = 1 \ldots k-1 \tag{9}$$

The latency of any node in the p'th segment is at least $\sum_{j=1}^{p-1} T_j$ and at most $\sum_{j=1}^{p} T_j$. Adding over all segments, we can sandwich OPT between two quantities:

$$\sum_{i=1}^{r-1} n_{>i} \cdot T_i \qquad \le \text{OPT} \qquad \le \sum_{i=1}^{r-1} n_{>i} \cdot T_i + \sum_{i}^{r} n_i T_i. \tag{10}$$

Now imagine doing the following in each segment except the last one: replace that segment by the shortest path that visits the same subset of nodes, while maintaining the starting and ending points (in other words, a traveling salesman path for the subset). We claim that the new latency is at most $(1+\epsilon)$OPT. Focus on the ith segment. The length of the segment cannot increase, and so neither can its contribution to the latency of nodes in later segments. The latency of nodes within the ith segment can only rise by $n_i T_i$. Thus the increase in total latency is at most

$$\sum_{i=1}^{r-1} n_i \cdot T_i. \tag{11}$$

Condition (9) implies that

$$\sum_{i=1}^{r-1} n_i \cdot T_i \qquad \le \epsilon \sum_{i=1}^{r-1} n_{>i} \cdot T_i,$$

which is at most ϵOPT by condition (10). Hence the new latency is at most $(1+\epsilon)$OPT, as claimed. (Aside: Note that we have thus shown that the lowerbound and upperbound in Condition (10) are within a $(1+\epsilon)$ factor of each other, once we ignore the contribution of the last segment.)

Of course, if we use a $(1+\gamma)$-approximate salesman path in each segment instead of the optimum salesman path in each segment, then the latency of the final tour is at most $(1+\gamma\cdot\epsilon+\gamma)$OPT.

5.3.1 The Algorithm

Combining the above ideas with those of Section 3, we obtain the following theorem.

Theorem 6 (Structure theorem) *There exist constants c, f such that the following is true for every integer $n > 0$ and every $\epsilon > 0$. For every well-rounded Euclidean instance with n nodes, a randomly-shifted dissection has with probability at least $1/2$ an associated tour that is $c \log n/\epsilon^2$-light, and whose latency is at most $(1+\epsilon)$OPT, where OPT denotes the latency of the minimum latency tour. The tour crosses each portal at most $f \log n/\epsilon$ times.*

Proof: Let $\mathcal{T}$ be the tour with minimum latency. As described above, we break it into $r = O(\log n/\epsilon)$ segments, where the ith segment has n_i nodes. We replace each segment except the last one by the optimum salesman path for that segment. This raises total latency by at most $\epsilon \cdot \mathrm{OPT}/4$, say. Now we lay down the randomly shifted dissection. Apply the technique of Section 3 in each segment, namely, to modify the segment so that it becomes portal-respecting and k-light where $k = O(1/\epsilon)$. (The last segment only has $\lceil 1/\epsilon \rceil$ nodes, so it is already k-light.)

A crucial observation is that the analysis of the tour modification in Section 3 relies on an expectation calculation, and so we can use *linearity of expectations* to analyse the cost of our $O(\log n/\epsilon)$ tour modifications.

The expected increase in the length of each segment is a multiplicative factor $(1 + \epsilon/4)$. Also, each salesman path never needs to cross a portal more than twice. We thus end up with a collection of paths which together are $O(k \cdot \log n/\epsilon)$-light (that is, $O(\log n/\epsilon^2)$- light) and do not cross any portal more than $O(\log n/\epsilon)$ times altogether.

As for the effect on the latency, note from (10) that the latency is sandwiched between two *weighted* sum of path lengths. Thus linearity of expectations implies that the expected increase in each weighted sum is at most a multiplicative factor $(1 + \epsilon/4)$. We conclude that with probability at least $1/2$, the increase in latency is a factor at most $(1 + \epsilon/2)$. Thus the overall latency of the final tour is at most $(1 + \epsilon)$OPT. $\square$

A simple dynamic programming running in $n^{O(\log n/\epsilon^2)}$ can compute the best tour satisfying the conclusion of Theorem 6; details are left to the reader. Note that the final segment with $n_r = \lceil \frac{1}{\epsilon} \rceil$ nodes can be guessed by exhaustive enumeration by trying all $n^{1/\epsilon+1}$ choices and then running the rest of the algorithm for each.

6 Survey of known results

All problems listed below are known to be NP-hard unless stated otherwise. A discussion of the problems circa 1995 appears in Bern and Eppstein [14].

6.1 Problems that have approximation schemes

Minimum Steiner Tree Discussed above.

k-median Discussed above.

Minimum Latency Tour Discussed above.

Facility Location We are given n nodes $\{x_1, x_2, \ldots, x_n\}$ who represent *clients* and m other nodes that represent potential *facilities*; each of the facility node has an associated cost c_j for opening a facility at that node. Each client gets "service" from the facility closest to it. The goal is to open a set of facilities S so as to minimize

$$\sum_{j \in S} c_j + \sum_{i=1}^{n} \min_{j \in S} \left\{ d(x_i, c_j) \right\}. \tag{12}$$

A variant of the problem involves specifying a *capacity* for each facility and a *demand* from each client. In the solution, the total demand from the clients assigned to a facility must not exceed the capacity at that facility. Many other variants exist. Aardal, Shmoys and Tardos [] give the first constant factor approximation; it works in any metric space. The metric space version is MAX-SNP-hard. Arora, Raghavan and Rao [9] give a PTAS for the geometric case. They extend the algorithm to the capacitated case but the final solution may violate capacity constraints by small amounts.

Generalized Steiner Problem Generalization of the minimum steiner tree problem in which p subsets $S_1, S_2, \ldots, S_p$ of the input nodes are specified and we have to find a Steiner forest such that nodes within each S_i are connected. (In the usual Steiner problem, $p = 1$.) Agarwal et al. gave the first constant-approximation for metric spaces. Our techniques give an approximation scheme whose running time is exponential in p, and which is a PTAS when p is sublogarithmic.

k-TSP Given n nodes and an integer k, find the shortest tour that visits at least k nodes. The TSP algorithm easily generalizes to k-TSP, although the running time is higher by a factor k [5].

k-MST Given n nodes and an integer k, find the shortest tree that visits at least k nodes. This problem was proved NP-hard not too long ago [26]. The approximation ratio for this problem has improved from $\sqrt{k}$ to $\log k$ to constant [19] to $1 + \epsilon$ [5].

Euclidean min-cost k-connected subgraph Given n nodes and an integer k, find the smallest subgraph that is k-connected. The subcase $k = 1$ is just the MST problem. Czumaj and Lingas [21] give a PTAS using techiques similar to those in our survey.

Prize collecting problems In *prize collecting TSP*, the input consists of a set of nodes and nonegative penalties on the nodes $\{\pi_i\}$. The goal is to find a tour on a subset of vertices that minimizes the sum of the cost of the edges in the tour and the penalties on the vertices not in the tour. One can design a quasipolynomial time approximation scheme using the methods of Section 2. The same is true for prize-collecting Steiner tree.

Min-cost perfect matching Given $2n$ nodes in the plane, we have to find the lowest cost set of edges that are vertex disjoint. This problem can be solved optimally in polynomial time. The techniques of Section 3 lead to a near-linear time approximation scheme. Recently Varadarajan [62] has found an $O(n^{1.5})$ time exact algorithm. His techniques are reminiscent of the techniques covered here, but more sophisticated.

Euclidean Max-Cut Given n nodes, find a partition into two subsets S_1, S_2 that maximises the sum of the lengths of the edges that have an endpoint in each of S_1 and S_2. Fernandez de la Vega and Kenyon [22] give a PTAS for this problem. The techniques are unrelated to those covered in our survey and also extend to any metric space.

Maximum traveling salesman The maximization version of the usual TSP —find the longest salesman tour visiting all n nodes— has been described in a lighter vein as the *frequent flier mileage maximisation problem*. Barvinok et al. [12] show that this problem has a PTAS. The idea is to approximate the unit ball by a polyhedron and then apply matching techniques and some partial enumeration.

6.2 Problems with no known approximation schemes

We suspect that many of the problems listed below may be MAX-SNP-hard.

Degree-restricted spanning tree. Given n nodes and a degree d, find the shortest spanning tree that has degree at most d; see Raghavachari [54] and Bern and Eppstein [14] for a discussion. The salesman path problem is a subcase when $d = 2$. Every minimum spanning tree has degree at most 5, so the problem is trivial for $d \geq 5$. The case $d = 4$ is NP-hard and the status when $d = 3$ is open. Khuller et al. [41] give 1.5- and 1.25-approximations for the two problems. The techniques of Section 2 seem very applicable but there is as yet no PTAS. (The author has certainly has tried to design one, and maybe others too.)

Vehicle Routing. This is really a large body of problems in operations research with several books devoted to them. The basic scenario involves a fleet of vehicles that have to make deliveries to customers. The vehicles have limited *capacities*, so they can only carry a limited number of parcels each. The vehicles may need to start from and end at a *depot* and the number of depots and their locations may be part of the input. The vehicles may be allowed to pick up packages in addition to dropping off packages. Clearly, many other variants can be defined. Constant factor approximations are known for many variants. Asano et al. [10] consider the most basic scenario: each package weighs the same, and each vehicle can carry at most k of them. Each customer receives a single package. The vehicles start from and finish at a central depot. Thus the problem can be rephrased as *minimum length covering by k-tours* (i.e., tours containing at most k nodes). This is somewhat reminiscent of capacitated k-median, which involves as a subcase *covering by k stars each of capacity n/k*. However, the difference in topology between the star and the tour seems to make the problem much harder. Asano et al. present a PTAS for the case $k = \Omega(n/\log n)$ (this uses techniques presented in this survey) and $k = O(\log n)$.

Minimum Weight Triangulation. Given a set of n nodes in the plane, the goal is to compute a triangulation that minimizes the Euclidean edge length. We do not know if this problem is NP-hard; it is one of the few problems on a famous list of 12 problems in Garey and Johnson [27] whose status is still open. Many candidate algorithms (such as Delaunay triangulation) give terrible approximations. Levcopoulos and Krznaric [] describe a constant factor approximation.

Minimum Weight Steiner Triangulation. This is a variant of the previous problem in which the triangulation may include any additional ("Steiner") points in the plane. We do not know if this problem is NP-hard. Eppstein has described a constant factor approximation algorithm for the problem, though this constant is fairly big (316, although this may be improveable).

Polygon Separation Given a collection of k polygons, separate them by a minimum-complexity planar straight-line graph. Edelsbrunner et al. [24] give a constant factor approximation for the case of convex polygons, and Mitchell and Suri extend this to arbitrary polygons.

Polyhedral Separation Given closed polytopes P, R in $\Re^3$ with $P \subseteq R$ we seek a polytope Q with the minimum number of faces such that $P \subseteq Q \subseteq R$. Brönniman and Goodrich give a constant-factor approximation. A related problem is *polyhedron approximation*: given a polytope R and a distance ϵ, find a minimum complexity polygon $Q \subseteq R$ whose boundary is within distance ϵ of the boundary of R. We do not know if this problem is NP-hard nor do we have a constant factor approximation.

Covering orthogonal polygons by rectangles. An *orthogonal polygon* is on whose all sides are horizontal or vertical. The polygon is simple if its boundary has a single connected component. A *rectangle covering* of a polygon is the minimum number of (possibly overlapping) rectangles whose union is the polygon. There is a constant factor approximation if the polygon is in general position —no two boundary segments are collinear—but only a logarithmic approximation (by reducing to Set Cover) for arbitrary simple polygons.

Graph Embedding. Given an n vertex graph G and a set of n nodes S in the plane, we wish to find a bijection from the vertices of G to S that minimizes the total embedded edge length. This problem generalizes the TSP (in which G is a cycle). Bern et al. [] give a $O(\log n)$-approximation when G is a tree. Similar approximations also exist for some other special cases.

TSP with neighborhoods. Given a collection of k simple polygons (not necessarily disjoint) with n nodes, we seek the shortest tour that passes through each polygon. The usual TSP is a special case in which each polygon is a point. Mata and Mitchell [47] describe a constant factor approximation.

6.3 Problems for which approximation schemes do not exist

k-center. Given n nodes $\{x_1, \ldots, x_n\}$ and an integer k, place k centers $c_1, c_2, \ldots, c_k$ in the plane so as to minimize $\max_{1\le i\le n} \min_{1\le j\le k} \left\{d(x_i, c_j)\right\}$. This is also called *minmax radius clustering*. Many heuristics give a factor 2-approximation; the first is due to Gonzalez []. Feder and Greene show that 1.82-approximation is NP-hard.

Covering nonsimple polygons with rectangles. This is a variant of a problem defined above; here the polygon may be nonsimple (i.e., have holes). Berman and Dasgupta show this problem is MAX-SNP-hard.

Polygon Bisection Given a simple polygon, partition it into two equal-area subsets using curved “fences” of minimum total length. No constant approximation ratio is achievable if $P \neq NP$ [43].

6.4 Geometric problems that resist our techniques

Obviously, this section is somewhat redundant because it should include every problem in Section 6.2. However, we discuss in some detail *why* our techniques have not made any headway yet. The problems just happen to be two that the author has thought about.

Covering with k-tours: We mentioned this simple version of vehicle routing before. In order to solve it by geometric divide and conquer, we seem to need a result stating that there is a near-optimum solution which enters or leaves each area a small number of times. This does not appear to be true. (More concretely, the Patching Lemma does not hold for this problem.)

We encountered a similar difficulty for the k-median problem (Section 5.2) but there we are able to restrict attention to portal-respecting solutions. Even though such a solution may enter dissection squares too many times, the "interface" between adjacent squares (i.e., amount of information that we have to decide upon so as to allow the algorithm to proceed independently in each square) is small. The interface can be specified by a logarithmic number of bits and so the dynamic programming can try all possible interfaces.

In the covering with k-tours problem, the difficulty lies in deciding upon a small interface between adjacent squares, since a large number of tours may cross the edge between them. It seems that the interface has to specify something about each of them, which uses up too many bits.

Minimum Weight Steiner Triangulation. At first blush this problem seems somewhat amenable to our techniques. One could try to define a portal-respecting solution (portals have a natural interpretation as Steiner points) and show that the best such solution has cost at most $(1+\epsilon)$OPT. Indeed, such a result was claimed in an early draft of [9] and then withdrawn.

The trouble with the approach arises from the topology of a triangulation. To convert an optimum solution into a portal-respecting solution, we deflect edges to make them pass through portals and use some kind of charging argument to show that the cost increase is small. There seems to be no obvious way to do this while keep the resulting structure a triangulation.

In fact, here is an open problem that tries to capture the difficulties we are referring to. Let $S(n)$ be the maximum number of Steiner points needed for the optimum triangulation on n nodes. (The maximum is over all —uncountably many—configurations of the n input nodes.) Is $S(n)$ finite? Bounded by a polynomial in n? Now let $S_\epsilon(n)$ be the analogous quantity for $(1+\epsilon)$-approximate triangulations. Is $S_\epsilon(n)$ finite? Polynomial in n for every fixed ϵ? Note that if the problem has a PTAS, then the answer to the last question has to be "Yes." Maybe showing a polynomial bound on $S_\epsilon(n)$ would be a first step towards the design of a PTAS. Note that a corollary of Eppstein's 316-approximation is a poly(n) (actually, $O(n \log n)$) upperbound on $S_{315}(n)$.

References

[1] F. Afrati, S. Cosmadakis, C. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the traveling repairman problem. *Informatique Theorique et Applications*, **20**(1):79–87, 1986.

[2] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein and M. Sviridenko. Approximation Schemes

for Scheduling to Minimize Average Completion Time with Release Dates. *Proc. 40th IEEE FOCS*, pp. 32-44, 1999.

[3] I. Althöfer, G. Das, D. Dobkin, and D. Joseph. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 1993.

[4] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. On the solution of traveling salesman problems. *Documenta Mathematica*, pp 645–656, 1998. (Extra volume, Proceedings of ICM.) *Preliminary version* Finding cuts in the TSP. Report 95-05, DIMACS, Rutgers University, NJ.

[5] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *JACM* **45**(5) 753-782, 1998. Preliminary versions in *Proceedings of 37th IEEE Symp. on Foundations of Computer Science*, 1996, and *Proceedings of 38th IEEE Symp. on Foundations of Computer Science*, 1997.

[6] 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.

[7] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. *Proc. ACM Symposium on Theory of Computing*, 1999.

[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *JACM* **45**(3):501–555, 1998. Prelim. version in IEEE FOCS 1992.

[9] 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.

[10] 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.

[11] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *JACM*, **41**(1), 1994. Preliminary version in IEEE FOCS 1983.

[12] A. Barvinok, S. Fekete, D. S. Johnson, A. Tamir, G. J. Woeginger, R. Woodroofe. The maximum traveling salesman problem. *Merger of two papers in Proc. IPCO 1998 and Proc. ACM-SIAM SODA 1998.*

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

[14] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In [34].

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] W. Fernandez de la Vega and C. Kenyon. A randomized approximation scheme for metric MAX-CUT. *Proc. 39th IEEE Symp. on Foundations of Computer Science*, pp 468–471, 1998.

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

[24] H. Edelsbrunner, A. D. Robinson, and X. Shen. Covering convex sets with nonoverlapping regions. *Discrete Math.*, **81**:153–164, 1990.

[25] 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.

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

[27] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman, San Francisco, 1979.

[28] 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.

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

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

[31] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pp 152-158, 1996.

[32] R. L. Graham. *Personal communication*, 1996.

[33] 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.

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

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

[36] 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.

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

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

[39] S. Khanna and C. Chekuri. A PTAS for Minimizing Weighted Completion Time on Uniformly Related Machines. *Proc. 28th ICALP*, pp. 848–861, 2001.

[40] S. Khanna and R. Motwani. Towards a Syntactic Characterization of PTAS. *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 329-337, 1996.

[41] 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.

[42] S. G. Kolliopoulos and S. Rao. A nearly linear time approximation scheme for the Euclidean k-median problem. *LNCS*, vol.1643, pp 378–387, 1999.

[43] E. Koutsoupias, C. Papadimitriou and M. Sideri. On the optimal bisection of the polygon. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pp. 198–202, 1990.

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

[45] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math*, **36**:177-189, 1979.

[46] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM J. Comp.*, **9**(3):615–627, 1980.

[47] 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.

[48] G. Miller. Finding small simple cycle separators for 2-connected planar graphs. *JCSS*, **32**:265–279, 1986.

[49] 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.

[50] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple PTAS for geometric k-MST, TSP, and related problems. *SIAM J. Comp.*, **28**, 1999. Preliminary manuscript, April 30, 1996. To appear in *SIAM J. Computing.*

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

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

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

[54] B. Raghavachari. Algorithms for finding low degree structures. In [34]

[55] 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.

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

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

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

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

[60] 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.

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

[62] K. R. Varadarajan A divide-an-conquer algorithm for min-cost perfect matching in the plane. *Proc. 39th IEEE Symp. on Foundations of Computer Science*, pp 320–329, 1998.

[63] V. V. Vazirani. Approximation Algorithms. *Springer Verlag*, Berlin, 2001.

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

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

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