Improved approximation schemes for geometrical graphs via "spanners" and "banyans"

Satish B. Rao & Warren D. Smith*
{satish,wds}@research.nj.nec.com

May 23, 1998

Abstract — **We give deterministic and randomized algorithms to find a Euclidean traveling salesman tour (TST) of length within $(1+1/s)$ times optimal. They run in $O(N \log N)$ time and $O(N)$ space for constant dimension and s. These time and space bounds are optimal in an algebraic computation tree model. We can also find a $(1+1/s)$ times optimal length 2-matching (M2M), edge cover (EC), minimum spanning tree (MST), Steiner minimal tree (SMT), rectilinear ditto (RSMT), and related graphs in the same time bound.**

This improves recent algorithms of Arora, which had used $N(\log N)^{O(s^{d-1})}$ time in fixed dimension d to produce a $(1+1/s)$ times optimal TST (or SMT, RSMT) with success probability 1/2. To verify success, however, Arora could only use a deterministic version of his algorithm that took a factor of N^d more time. The increase in running time for our deterministic version depends only on s.

Arora's approach can also be extended to produce other $(1+\epsilon)$-approximate geometrical graphs besides TSTs, e.g. the minimum matching (MM) and subgraph versions of the MST and TST problems. Our methods (at least at the moment) don't apply to those problems, but we can produce a $1.001 \exp(8 \cdot 2^{1-1/(d-1)}\sqrt{d})$-approximate MM in $O(N \log N)$ Monte Carlo time.

Our algorithms are based on using low-weight Euclidean *spanner graphs* (and generalizations of them) in conjunction with the hierarchical structure theorems that serve as the basis of Arora's work.

A "t-spanner" is a graph on N sites such that shortest path distance between two spanner vertices approximates the Euclidean distance to within a factor of t. By making work by Arya, Das, et al. more explicit, we show that a $(1+1/s)$-spanner of N sites in d-space is computable in $(sd)^{O(d)} N \log N$ time and $(sd)^{O(d)} N$ space, has maximum valence $(sd)^{O(d)}$, and is $(sd)^{O(d)}$ times longer than the MST. We show that spanners can in principle be made orders of magnitude shorter and simpler by allowing "Steiner points."

We introduce a remarkable generalization of the notion of "t-spanner," the "t-banyan," that t-approximates interconnection costs for site subsets of *any* cardinality, not just 2. If $d \ge 1$ and $\epsilon > 0$ are fixed, we show a $(1+\epsilon)$-banyan of N sites in d-space exists, has $O(N)$ vertices and $O(N)$ edges, is only a constant factor longer than the MST, and is computable in $O(N \log N)$ time.

Keywords — Polynomial time approximation schemes, optimal algorithms, derandomization, Traveling salesman tour, Steiner minimum tree, Minimum spanning tree, Minimum matchings, 2-matchings, Edge cover, Rectilinear Steiner minimum tree, quadtrees, spanners, banyans.

CONTENTS

*NECI, 4 Independence way, Princeton NJ 08544

1 Introduction

We improve a method of Arora [1] for approximating TSTs and SMTs. Arora's algorithm, given N points in a Euclidean d-space, $d \geq 2$, produces, with probability $1/2$, a $(1+1/s)$-approximate TST. It runs in $N(\log N)^{O(s\sqrt{d})^{d-1}}$ time ($s > 1$). However, the algorithm can fail, and these failures are not easy to detect. The only options Arora provided were either to hope failure did not occur[1] or to use a derandomized version of his algorithm which takes N^d times longer.

Our improved methods yield $O(N \log N)$ time algorithms for this problem for any fixed s and d. In more detail, we obtain when $d = 2$

- an algorithm that finds a $(1 + 1/s)$ times optimal TST (or SMT, RSMT, MST, EC, M2M) with probability $1/2$ in
$$s^{O(s)}N + O(dN \log N) \tag{1}$$
Monte Carlo time and
$$s^{O(1)}N + s^{O(s)} \log N \tag{2}$$
space,

- or deterministic and Las Vegas algorithms that run in
$$2^{s^{O(1)}}N + s^{O(1)}N \log N \tag{3}$$
time and consume
$$s^{O(1)}N + 2^{s^{O(1)}} \log N \tag{4}$$
space.

In d dimensions, our methods yield algorithms that succeed with probability $1/2$ in Monte Carlo time
$$(s\sqrt{d})^{O(d(\sqrt{d}s)^{d-1})}N + O(dN \log N) \tag{5}$$
and
$$(sd)^{O(d)}N + (s\sqrt{d})^{O(d(\sqrt{d}s)^{d-1})} \log N \tag{6}$$
space, or in Las Vegas time
$$2^{(sd)^{O(d)}}N + O(dN \log N), \tag{7}$$
or a deterministic algorithm with runtime
$$2^{(sd)^{O(d)}}N + (sd)^{O(d)}N \log N, \tag{8}$$
in both of the latter cases consuming
$$(sd)^{O(d)}N + 2^{(sd)^{O(d)}} \log N \tag{9}$$
space.

In fixed dimension d and with fixed s, our approximately optimal TST (or MST, SMT, RSMT, M2M, EC) algorithms run in $O(N \log N)$ time and $O(N)$ space.

Our $(1 + 1/s)$-approximation algorithms for NP-hard problems (e.g. TST) run more slowly than competing exact algorithms when $s^{O(1)} > N$ or $d^{O(1)} > \log N$, otherwise they run more quickly. For problems in P (e.g. M2M) we run more slowly than competing exact algorithms when $s^{O(1)} > \log N$ or $d^{O(1)} > \log \log N$, otherwise we run more quickly.

Our algorithms use low-length Euclidean *spanner graphs* in conjunction with the hierarchical structure theorems that underly Arora's work.

Definition 1 *A "$(1+\epsilon)$-spanner" of a set of points is a subgraph of the complete Euclidean graph where for any u and v the length of the shortest path from u to v is at most $(1+\epsilon)$ times the Euclidean distance between u and v.*

Das, Narasimhan & Salowe [10] showed that spanners could be found with sum of the edge lengths at most $O(a_d(\epsilon)\ell(MST))$ where $\ell(MST)$ is the length of the minimum spanning tree of the sites. They did not specify the function $a_d(\epsilon)$. We will show in theorem 45 that $a_d(\epsilon) = (d/\epsilon)^{O(d)}$.

We also study extending the notion of Euclidean spanner graphs to allow Steiner points. Specifically, we prove in §6.1 that Steinerized $(1+\epsilon)$-spanners can be as much as order $\epsilon^{-1}/|\log \epsilon|$ times shorter than any ordinary $(1+\epsilon)$-spanner in 2D, and $\epsilon^{-\Omega(d)}$ times shorter in any fixed dimension d. On the other hand, we show in theorem 56

[1] k repeated failures is an event of probability $\leq 2^{-k}$ *if* your random number generator is truly random and your code is bug free.

that in fixed dimension d the bound $a_d(\epsilon) = (d/\epsilon)^{O(d)}$ is in fact optimal (up to the inherent weakness arising from an "$O(d)$" in the exponent) even if Steinerized spanners are allowed.

Definition 2 *A "$(1+\epsilon)$-banyan" of a set of sites is a graph (whose edges are line segments) such that for any subset S of the sites, the length of the shortest connected subgraph of the banyan which includes S, is at most $(1+\epsilon)$ times longer than the Steiner minimal tree of S.*

Banyans are introduced for the first time in this paper. We show in §4 that for fixed ϵ and d, a $(1+\epsilon)$-banyan of any set of N sites in d-space exists, which

- Is at most a constant factor longer than the MST of the sites,
- Has $O(N)$ Steiner points,
- Has all valencies bounded,
- Is computable in $O(N \log N)$ time.

1.1 *Definitions of the geometrical graphs (and their acronyms!)*

Given N point sites in Euclidean space, a "geometrical graph" is a set of line segments ("edges") whose endpoints include the sites.

The optimum *traveling salesman tour (TST)* is the shortest cyclic path that visits every site. A $(1+\epsilon)$-*approximate TST* is a TST with length at most $1+\epsilon$ times the length of the optimum TST.

A *Steiner tree* is a network which connects all the sites. If the network consists solely of line segments parallel to the coordinate axes, then it is a *rectilinear Steiner tree* (RST). The *Steiner minimal tree* (SMT) is the shortest Steiner tree, and the RSMT is the shortest rectilinear Steiner tree. (Alternatively, you can think of the RSMT as a Steiner tree where we measure length in the L_1 metric. With that view, the line segments in RSMT can have arbitrary slopes.) A *spanning tree* is a network made of site-to-site line segments *only*, which connects all the sites. The *minimum spanning tree* (MST) is the shortest spanning tree. A *matching* is a partitioning of the N sites into $N/2$ pairs (for this, we assume N is even). The *minimum matching* (MM) is the matching such that the sum of the lengths (i.e. distances between the 2 members of) of the pairs is minimum. The *all nearest neighbor* digraph (ANND) is the digraph in which every site is joined to its nearest neighbor; ANN is the undirected version of this graph (with duplicate edges uniquified). To define the *directional nearest neighbor* graph we need a covering of the sphere of possible directions by spherical caps of some angular radius θ (for the "θ-DNNG"). The DNNG edges are bonds from each site to each of its nearest neighbors within each cone of directions corresponding to a cap in the covering. An *edge cover* is a set of edges containing every vertex at least once; the minimum edge cover (EC) graph is the minimum length graph in which every valency is ≥ 1. The minimum 2-matching (M2M) is the minimum length graph in which every valency is exactly 2 (and doubled edges are permitted in the graph). The *sphere of influence* graph (SOI) is defined as follows. Associate with every site v a ball B_v centered at that site with radius the distance to the site's nearest neighbor. There is an edge ij in SOI if B_i and B_j intersect.

Finally, the *minimum bipartite matching* (MBM) among N red and N blue points [31] is the minimum length pairing of each red point with a unique blue mate.

Properties of these graphs will be discussed later; for now we merely mention that in any metric space,

$$\ell(SMT) \leq \ell(MST) \leq \ell(TST) \leq 2\ell(SMT); \tag{10}$$

$$\ell(EC) \leq \ell(ANN) \leq \ell(ANND) \leq 2\ell(EC) \leq 2\ell(MM) \leq \ell(TST). \tag{11}$$

$$\ell(EC) \leq \ell(M2M) \leq 2\ell(MM) \tag{12}$$

$$M2M \subseteq SOI; \;\; ANN \subseteq MST. \tag{13}$$

It is possible to find MST, ANN, ANND, EC, M2M, SOI, and MM of N points in d-space (or with an arbitrary distance matrix) in polynomial time, but finding the optimal TST, SMT or RSMT is NP-hard. Inequalities (10-13) immediately lead to factor-2 approximation algorithms for TST and SMT in any metric space, however.

1.2 *Random versus nonrandom*

One kind of randomness in algorithms consists of making probabilistic assumptions about the input, e.g. that the N sites were sampled from a uniform distribution in a d-cube. Ideas of this nature, although providing very useful guidelines in practical programming, should *not* be relied on[2]. Both we and [1] avoid any probabilistic assumptions about our input, and instead derive all our randomness from a random number generator.

Another kind of randomness – the kind in [1] – is "Monte Carlo" algorithms. These are algorithms which succeed with some probability (say 1/2), or fail – but there may be no easy way to distinguish success from failure. Arora [1]'s algorithms may also be regarded as running in some worst case runtime bond and *never* failing – if we change their task to be that of providing a tour whose length excess (versus the optimal tour) is a random variable whose *expectation value* is $\leq \epsilon\ell(TST)$ – but there is no easy way to tell whether we are greatly exceeding the expectation value. Many people do not like Monte Carlo algorithms because it is difficult to debug them (is my failure a bug or a feature?) and one can never be sure when to stop running them.

[2]For example, Fortune's "sweepline" algorithm for computing 2D Delaunay triangulations [16], although in theory deterministic and featuring $O(N \log N)$ runtime in the worst case, was *implemented* by Fortune (in a widely distributed C program) without tree balancing. The resulting code exhibited N^2 runtime for N points forming a convex N-gon.

A better kind of randomness is "Las Vegas" algorithms. These algorithms (in flavor A) succeed, in some worst-case runtime bound, with some probability (say $1/2$), or fail, but when they fail, they say so. Flavor B (the two flavors are easily interconverted, of course) never fails but its runtime bound is only a bound on the *expected* running time[3].

Best of all are fully deterministic algorithms with worst case runtime bounds.

In this paper we will provide all[4] algorithms in all three varieties (Monte Carlo, Las Vegas, and fully deterministic). Our Monte Carlo algorithms are the fastest. Our fully deterministic algorithms are slower than our Las Vegas ones, but not by enough that the difference is visible in our worst case time bounds.

1.3 How close are we to optimal speed?

The $N \log N$ behavior of our TST, MST, SMT, EC, ANN, M2M, MM, spanner, banyan, or RSMT runtime (8) when s and d are held fixed is optimal in an "algebraic computation tree" model [9][5]. Although our algorithms are most easily expressed in a way that use the "floor" function (so they are not pure algebraic, and hence initially would seem not to be susceptible to the lower bound) they may be rewritten (§1.5) in an honestly purely algebraic manner still with $O(N \log N)$ runtime.

It is very likely that (8)'s dependence on s with d held fixed is optimal. The known NP-completeness proofs for 2D TST (or RSMT, SMT) involve points on a polynomial size grid in which all TST edges are in a fixed number of possible orientations. If s were polynomially (in N) large, then a $(1 + 1/s)$-approximation would be exact. Assuming NP-hard problems aren't soluble in $2^{N^{o(1)}}$ time, implies that it would be impossible for our runtime to be $2^{s^{o(1)}} N^{O(1)}$ with $d = 2$. But in fact our runtime bounds are the same as this "impossible" result but with the o changed to an O.

Finally, we can make it plausible that the dependence of our TST running time on d with s held fixed is close to optimal. Trevisan [41] showed that approximating TST to within any sufficiently small constant factor for N sites in $\log N$ dimensional space, is NP-hard[6]. Hence (unless NP-hard problems are soluble in $2^{N^{o(1)}}$ time) no algorithm for approximating TST to within any sufficiently small constant factor for N sites in d-space can run in $2^{2^{o(d)}} N^{O(1)}$ time. We get $2^{d^{O(d)}} N^{O(1)}$ so that our dependence on d is also tight *except* that we have a $d^{O(d)}$ instead of a $2^{O(d)}$. It's likely that *this* improvement is actually possible.

1.4 Applicability

Roughly speaking, our methods apply to a geometrical graph G if it's sparse, simply defined[7], there are only $r^{O(r)}$ possible kinds of "boundary conditions" if r is the number of crossings of G over a hyperplane, G obeys a "patching lemma" (cf. §2), the site coordinates may be rounded to integers of order $(Ns)^{O(1)}$ without ruining the approximation, and G can't be more than a constant factor shorter than the MST as $N \to \infty$ in fixed dimension.

TST is the perfect good example. A bad example of a graph – which neither our methods nor Arora's can handle – is the minimum bipartite matching (MBM), because it has no patching lemma. Similarly, *maximum* length spanning trees, TSTs, etc. also can't be handled.

Arora's methods had also produced approximation schemes for minimum length matching (MM) and k-vertex subgraph variants of the minimum spanning tree and TST problems in Euclidean space. Our methods (at least currently) don't apply to these graphs because they can be much shorter than the MST and the coordinates are not roundable.

However, with sufficient cleverness, for any particular graph it *might* be possible to overcome its shortness and non-roundability. To illustrate that, we introduce the minimum edge cover graph EC, and the minimum 2-matching M2M, both of which can be much shorter even than MM, but which we nevertheless (§4.3) can approximate to within $1 + 1/s$ with our usual time and space bounds. This gives some hope that it might be possible to handle MM in the future, although we do not know how at present. In §4.4 we show that our techniques nevertheless can approximate MM to within a factor of $2^{O(\sqrt{d})}$ in $d^{O(d)} N \log N$ *Monte Carlo* time.

1.5 Computational model

Our time and space bounds will be in the usual "real RAM" model in which arithmetic operations $(+, -, \times, \div, \sqrt{\ }, >, =)$ or IO for real numbers take unit time. Also, a real number may be stored in 1 memory location. Occasionally we will assume that the input points are in "general position," that is, their coordinates don't obey any nontrivial algebraic equations. This may

[3] "1-sided hybrid" algorithms are also possible: The "strong pseudoprime test" is Las Vegas for proving compositeness, but only Monte Carlo for verifying primality. The literature also contains the alternative related terminology (which we will avoid) of "BPP" and "RP" algorithms.

[4] With one exception, the Monte Carlo matching algorithm of §4.4.

[5] Actually [9] did not prove this for spanners and banyans. However by applying any $o(N \log N)$ MST algorithm [6] to either of these graphs we would recover an approximate MST, hence by [9]'s $\Omega(N \log N)$ lower bound for approximate MSTs, the result follows. They also did not prove this for MM, M2M, EC, ANN, but we do claim this (although we omit the proof). We also claim that approximating the shortest pair distance to within any constant factor requires $\Omega(N \log N)$ time. All these $\Omega(N \log N)$ bounds apply even in 1D. M.Smid points out that [13] also proves $N \log N$ lower bounds for computing spanners, and [38] shows an $N \log N$ lower bound for approximate min weight matching.

[6] Also: approximating RSMT to within any sufficiently small constant factor for N sites in $N - 1$ dimensional space, is NP-hard.

[7] SMT and RSMT aren't very "simply defined" since they involve vertices which are not input sites, which is an obstacle we had to overcome using some new ideas.

be assured either by a random pre-perturbation or by self-consistent tie-breaking schemes.

We will sometimes assume that $s = O(N)$, because it simplifies some of our bounds, and one might as well because demanding more accuracy than this from our approximation algorithms will make them run more slowly than exact algorithms.

Although some steps in our algorithms are most conveniently accomplished with the aid of the "floor" function $\lfloor x \rfloor$, we claim this isn't really needed[8]. For TST, RSMT, and SMT, really all we need are $O(\log N)$ bits of precision in a word, but MM, M2M and EC could require arbitrary precision (if the points come as a large number of very close pairs).

1.6 Quick sketch of our methods

In this section we sketch our methods in the context of the Euclidean plane. Again, many of the techniques were previously developed by Arora in [1].

We assume that the sites are on integer coordinates of an $L \times L$ grid. We then choose random integers a and b from $[0, L]$. We grow the width of the grid by L by extending it on the left by a grid lines and on the right by $L - a$ grid lines. We similarly grow its height using a random variable b.

Our algorithm begins by finding a $1 + O(1/s)$-spanner of the set of input points which is only s^K times longer than MST for some constant K. An optimal TST T_S in the spanner has length within $1 + O(1/s)$ of the optimal tour T of the input points. We also note that an optimal TST in the spanner only uses any edge of the spanner twice.

Then we form a quadtree decomposition of our randomly grown square containing the sites as follows. We first subdivide the square into four equal sized squares, and recurse on the subsquares with more than one site inside. This takes $O(N)$ time per quadtree level, and it turns out we can require our points to lie on a grid with $L = O(Ns)$ lines, so the quadtree will have $O(\log(Ns))$ levels.

We will use a sequence of "patchings" to modify the spanner with respect to the quadtree decomposition. A patching essentially cuts some edges in the spanner at square boundaries and *adds three line segments* that reconnect all of the cut edges.

We apply a sequence of patchings to the spanner S so that

1. The total length of the added line segments is $1/r$ times the original length of the spanner,
2. and the boundary of each quadtree square is intersected by the edges of the resulting graph at most $O(r)$ times.

The procedure above along with its analysis is derived easily from the proof of Arora's main structure theorem in [1].

We set $r = O(s^{K+1})$ in the procedure above. This produces a modified graph S' that consists of the line segments from S plus some additional line segments whose total length is $O(s^K MST)/r = O(MST/s)$.

A tour exists in S' which is only an additive amount $2O(MST/s)$ longer than the best tour in the unmodified spanner, as may be seen via the following argument. Follow the tour T_S from the spanner but alter (by deleting certain 2-way line segment crossings) the resulting tour so that it only uses each additional line segment in the modified spanner at most twice. This tour T'_S has length that is at most an additive amount $2O(MST/s)$ larger than T_S. Since MST is at most the cost of the optimal tour, we can conclude that there is a tour in the modified graph of cost at most $1 + O(1/s)$ times optimal.

Moreover, the modified spanner has a recursive decomposition, induced by our quadtree, in which each piece only interacts with the outside a constant r times. This allows us to use dynamic programming to find the optimal TST in the modified spanner in time that is exponential in $r = O(s^{K+1})$ but linear in N. (The same standard techniques were used by [1]).

We remark that, unlike Arora [1], we can efficiently derandomize the procedure above. This is due to the fact that Arora considered applying his modification procedure to some unknown solution to the TST problem. We actually use the modification procedure on the spanner graph which we explicitly compute. Thus, we easily check for the success of Arora's procedure. Indeed, we can give an algorithm to derandomize it completely running in added time $O(N \log N)$.

Arora's arguments had shown that, with probability $1/2$, there is an $(1 + 1/s)$-approximate TST that only crosses any square's boundary in a random quadtree decomposition at most $O(s)$ times.

Since $O(s)$ is significantly less than $q = O(s^K)$, we can find a faster dynamic programming algorithm that with probability $1/2$ finds a $(1 + 1/s)$-approximate tour, and runs in time $O(q)^s N = O(s)^{Ks} N$. This can be compared with our deterministic dynamic programming algorithm's runtime of $O(q^q)N = O(s)^{Ks^K} N$. But our faster randomized algorithm is Monte Carlo; like Arora's randomized scheme, it can fail without any indication that it failed.

1.7 Related Work on Geometrical Graph algorithms

Christophides [7] observed that by finding the minimum spanning tree of the graph, and then adjoining the minimum length matching of all the odd-valent points in the spanning tree, an Eulerian (even-valent) graph CG resulted, and in any metric space $\frac{3}{2}\ell(TST) \le \ell(CG)$. In

[8] In an early step of our TST algorithm we scale all coordinates and then round them to integers of order N. This may be accomplished either by Nd invocations of the floor function, or by $O(Nd \log N)$ operations without resorting to floor. It is often convenient to have available coordinates scaled and rounded to integers of order $2^{-k}N$ for all k. These quantities may be pretabulated in $O(Nd \log N)$ non-floor operations. For what might happen if we allow a more powerful computational model, see §5.2.

fact $\frac{3}{2}\ell(LP) \le \ell(CG)$, where $\ell(LP)$ is the cost of a simple linear programming lower bound on the TST cost (also computable in polynomial time) [37]. This approximation ratio 1.5 is still best known for TST in arbitrary metric spaces for approximation algorithms running in polynomial time.

Smith [39] showed how, by "searching over separators," to find the optimum TST for N points in the Euclidean plane in $N^{O(\sqrt{N})}$ steps. (Other geometrical graphs are calculable with the same technique, for example the minimum length triangulation.) Also in [40] he extended this to handle RSMT.

Smith [39] showed how to find a θ-DNNG of N sites in the plane in time $O(N \log N)/\theta$; see also our theorem 46.

Vaidya [43] showed how to find ANND in $O(d)^d N + O(dN \log N)$ time by a tree-of-boxes algorithm[9].

The SOI graph is computable in $O(N \log N)$ time when $d = 2$ by computing the ANN spheres from the Delaunay triangulation and then finding their intersections (of which there will be[10] $2^{O(d)}N$) via the "plane sweep" paradigm [39]. In higher dimensions d, one may use Vaidya's algorithm to find the ANN spheres and then find their intersections with techniques of [32] and [15]. The resulting algorithm will run in $d^{O(d)}N + O(dN \log N)$ steps. This obsoletes the previous best runtime $NO(\log N)^{d-1}$ of [39].

A $(1 + 1/s)$-approximate MST may be found by computing a $(1+1/s)$-spanner of the N sites, and then finding the minimum spanning tree lying inside the spanner in time $O(N + E_S)$, where E_S is the number of spanner edges, using the linear time MST algorithm of [24]. The resulting runtime (cf. theorem 45) is $(sd)^{O(d)}N + O(dN \log N)$ in space $O(s^d N)$. When $d = 2$ the exact MST may be found in $O(N \log N)$ steps by taking advantage of known $O(N \log N)$-time algorithms to find the "Delaunay triangulation" DT and using the fact that $MST \subset DT$.

Similarly, a $(1 + O(1/s))$-approximate MM may be found by finding the minimum length "odd valent subgraph" (OVS) inside a $(1+1/s)$-spanner graph, and then converting the connected components of the OVS into matchings by considering the shorter of the two alternating matchings inside a (shortcut) cyclic tour round their minimum spanning trees. Vaidya in section 6.1 of his paper [42] shows how to find a min-length odd-valent subgraph in a V-vertex E-edge spanner graph by solving a min-weight matching problem in an easily generated associated graph (in which a full matching must exist) with $\le 2E + V$ vertices and $\le 5E + 5V$ edges. The resulting[11] runtime is $O(s^{1.5d}N^{1.5}(\log N)^{1.5}\alpha(N, N))$ in space $O(s^d N)$. If $d = 2$, the d's in the bound above may be replaced by 1's by using a DNN graph as one's spanner (see §6 and [39]) instead of the spanners arising from [5]. Still better dependence on s is perhaps achievable.

The "min-weight degree constrained subgraph (DCS) problem" is: inside an edge-weighted graph with V vertices and E edges, we ask for a subgraph having minimal total weight whose valency at each vertex v is at least L_v and at most U_v. (We specify the arrays L and U.) At the end of [18], it was remarked that this problem could be solved (provided all the edge weights were non-negative integers $\le C$) in time $O(\sqrt{V\alpha(E, V) \log C} E \log(VC))$, but the details were deferred to a "forthcoming paper" and also to a second manuscript – and unfortunately neither was ever published. Since our EC graph is the special case of this problem that arises when $L_v \equiv 1$, $U_v \equiv V - 1$, the weights are Euclidean distances and the graph is the complete graph, we see that it is computable in polynomial time. Similarly for M2M.

If the reader is unhappy about relying on unpublished results of Gabow and Tarjan [18], we remark that it's made clear how and why the DCS problem is soluble in polynomial time, and in fact is reducible to a min-weight matching problem, in chapter 10 of [30]. Indeed, [30] even mentions graph transformations which will allow the solution of, for example, the parity constrained DCS problem (PDCS) – the valence of v is constrained to be an integer between L_v and U_v with parity P_v; also some vertices's parity may be left unconstrained. Unsatisfiable sets of valence constraints will be detected.

We now claim that Vaidya's approximate minimum matching algorithm should be generalizable to an algorithm to produce a $(1 + 1/s)$-approximation for the Euclidean degree constrained subgraph problem (for *any* degree constraints)[12]. This is accomplished by solving a PDCS problem in the spanner graph using the correct lower bounds L_v (but infinite upper bounds U_v), and then shortcutting paths to single edges as necessary to get the valencies down to the true U_v.

The remarks above yield the best "previously known[13]" algorithms for approximating EC and M2M. However, they are obsoleted by the algorithms of this paper if d and s are sufficiently small.

Apparently the current record polynomial time approximation factor for SMTs in arbitrary metric spaces is ≈ 1.644, due to Karpinski & Zelikovsky in an unpublished manuscript. SMT in a general metric space is at least $\frac{r2^r+s}{(r+1)2^r+s}$ times as long [4][14] as the shortest union

[9] Vaidya only claimed $O(d)^d N \log N$, but a careful examination of his algorithm shows this more precise bound is also valid.

[10] One may prove this by showing that the valency of the site with the shortest distance to its nearest neighbor is $2^{O(d)}$. Then mentally remove this site and generate the SOI graph of the remaining sites; the result will be a supergraph of the SOI graph with $2^{O(d)}N$ edges.

[11] This slightly improves Vaidya's runtime and reduces his space needs down to linear, because we've used the spanners arising from [5] instead of Vaidya's spanners, and we've used the $O(\sqrt{V\alpha(E, V) \log V} E \log(VC))$-time matching algorithm of [18].

[12] And it will run in the same time bound as Vaidya's plain approximate Euclidean minimum matching algorithm, if one believes that the time bound in the unpublished Gabow-Tarjan result will apply for the ODCS problem.

[13] They weren't really previously known, since we just pointed them out for the first time in print.

[14] This result does not lead to a "polynomial time approximation scheme" for SMTs in general metric spaces. In fact one does not exist unless P=NP, since the problem is known to be MAX-SNP

of k-site SMTs with $k = 2^r + s$, $0 \le s < 2^r$; RSMT in the Euclidean plane is at least $2/3$ times as long as the L_1 MST [23]; SMT in the Euclidean plane is at least $\sqrt{3}/2$ times as long as the MST [12]; and all these factors are tight.

However in arbitrary metric spaces it is known that approximating SMT or TST to within $1+\epsilon$ is NP-complete, for all sufficiently small constant $\epsilon > 0$. (See [1] for the references to this recently developed area of "MAX-SNP hardness.")

2 Approximating the TSP in the plane

We use lemmas stated in [1] and [25] about $(1+\epsilon)$-approximate traveling salesman tours for N sites in the plane. They all generalize to d-space with $d > 2$, but for now we will state some of them only in the case $d = 2$.

Lemma 3 (Perturbation Lemma)
Given a graph G on N input sites in $[0,1]^d$, one can perturb the sites so that they are on points of the form $(i/k, j/k)$ for integers i, j and with common denominator k, so that the total length of the graph changes by an additive term of at most $\sqrt{d}E_G/k$ where E_G is the number of edges in G.

Assuming WLOG that the point set in $[0,1]^d$ has MST length $M \ge 1$, by using the lemma above and scaling by a factor of $L = \sqrt{d}N/(\delta M)$, we may alter a traveling salesman problem so that its sites lie on integer points in $[0,L)^d$. This introduces an additional $(1+\delta)$ factor to the final TST approximation factor, which we will ignore in the following for simplicity. We can then assume the following invariant on a traveling salesman problem for the rmaineder of this section.
Rounding Assumption: *A $(1+\delta)$-approximate traveling salesman problem has N sites with integer coordinates in $[0,L)^d$, $L \le \sqrt{d}N/\delta$, such that their MST has length $\le \sqrt{d}N/\delta$.*

We assume that the sites are on integer coordinates of an $L \times L$ grid. We then choose random integers a and b from $[0,L]$. We grow the width of the grid by L by extending it on the left by a grid lines and on the right by $L-a$ grid lines. We similarly grow its height using a random variable b.

Definition 4 *We define $t(G,l)$ for a graph G and a line segment l in $\Re^2$ to be the number of times any edge in G crosses l.*

We also state a version of a lemma from [1].

Lemma 5 *Given a graph G in $\Re^2$ on N sites*

$$\sum_{l:\text{horizontal}} t(G,l) + \sum_{l:\text{vertical}} t(G,l) \le \sqrt{d}\ell(G), \qquad (14)$$

where $\ell(G)$ denotes the length of G, and l are lines (in d-space, hyperplanes) of the integer grid.

hard.

The above lemma (proven in [1]) follows from the fact that the L_1 and L_2 norms differ by at most a $\sqrt{d}$ factor.

Definition 6 *A crossing between a G-edge and a quadtree box boundary is "relevant" if exactly one of the endpoints of the G-edge is inside the box.*

Lemma 7 *If $\ell(G) = k \cdot \ell(MST)$, then the total number of quadtree-G crossings for a set of sites meeting the Rounding Assumption is $\le dkN/\delta$. Even without assuming the Rounding Assumption, the number of relevant crossings is $O(E_G \log_2(2L))$ for a $\log_2(2L)$-depth quadtree hierarchy where E_G is the number of edges in G.*

Definition 8 *A TST (or general graph G) is "r-light" if it crosses each (hyperplanar) boundary between two sibling quadtree boxes at most r times. It is "r-vapid" if each such boundary is crossed at most r times by relevant crossings. (r-light $\Rightarrow$ r-vapid.)*

Definition 9 *A "(g,κ,q)-patching procedure" takes a graph G in $\Re^2$ and a line segment l of length L and produces a new graph G' with total additional length gL that crosses the line segment at most κ times. Moreover, G' consists only of line segments of G and subsegments of $\le 2q$ infinitesimally shifted versions of l, $\le q$ of them on each side of l.*

Theorem 10 *Given a graph G on a 2D grid, a (g,κ,q)-patching procedure for the graph and line segments in $\Re^2$, and a quadtree dissection with a random shift (a,b), then one can use a sequence of patching procedure calls to produce whichever is desired of the following:*

- *An $(r+2q)$-light graph G', for any r with $r \ge \kappa$, that is in expectation at most $(1+\frac{2g\sqrt{2}}{r+1-\kappa})^2$ times longer than G.*
- *An r-light graph G', for any r with $r \ge \kappa$, that is in expectation at most $\exp\frac{4g\sqrt{2}}{r+1-\kappa}$ times longer than G. Note: for this variant we need a slightly different definition of "r-light" arising from a slightly different (now binarized) notion of what the "levels" are in the quadtree; see the end of the proof.*

In other words, if g, q, and κ are fixed, the length increase factor incurred when modifying G to make it r-light is $1+O(1/r)$.

Proof. This is our version of Arora [1]'s "structure theorem." Our proof is essentially the same as his, but:

1. We have a more general theorem statement,
2. We will be more careful about our constant factors, in an attempt to get better and more explicit bounds,
3. A new trick allows us to extend the domain of applicability of the theorem as far as possible, e.g. allowing 1-lightness when $\kappa = q = 1$.

We will patch the graph to make it r-light with a sequence of calls to the patching procedure on line segments that are subsegments of the horizontal and vertical lines of the quadtree dissection. We will show that for each line l in the quadtree dissection the expected (over the random choice of (a,b)) increase in the length due to patching procedures involving subsegments of l is bounded by $O(g\kappa s/r)$. By combining this with the fact that $\sum_{l\in Q} t(G,l)$ is bounded by the length of G and by using linearity of expectation, the theorem will follow.

We discuss only vertical lines for now.

For a line, we define its "level" to be the level of the largest box for which part of it serves as a border. (This is the smallest number $i \geq 0$, such that the line borders a level i box.) Note that the probability for a line to have level i with a random shift (a,b) is

$$\leq \frac{2^i}{L}. \qquad (15)$$

We patch the graph by using the following procedure on each line. Recall that the actual shifted dissection consists of lines of length $2L$.

ARORA-MODIFY(l,i,b)
(l is a vertical square border in the quadtree dissection and b is the vertical shift of the dissection, and i is the level of line l given b.)
For $j = 1+\log_2 L$ down to i do:
For $p = 0,1,\ldots,2^j-1$, if the segment of l between the y-coordinates $b+p2^{-j}L$ and $b+(p+1)2^{-j}L$ is crossed by the current G more than r times, then use the patching procedure to reduce the number of crossings to κ.

For $j \geq i$, define $c_{l,j}(b)$ to be the number of edges to which the patching procedure is applied during the jth iteration of the outer for loop in ARORA-MODIFY(l,i,b). Since each patch replaces $\geq r+1$ crossings by κ crossings, we have for each vertical shift b

$$\sum_{j\geq 0} c_{l,j}(b) \leq \frac{t(G,l)}{r+1-\kappa}. \qquad (16)$$

Moreover, by the definition of a (g,κ)-patching procedure the increase in length of the resulting graph caused by ARORA-MODIFY(l,i,b) is at most

$$g\sum_{j\geq i} c_{l,j}(b)\cdot\frac{L}{2^{j-1}}. \qquad (17)$$

Recall that for a line l the level is i with probability at most $2^i/L$ (over the choice of the horizontal shift a). Thus for every vertical line l and every b, $0 \leq b < L$, the expected increase in cost due to applications of the patching procedure to its subsegments is bounded by

$$\begin{aligned} g\sum_{i\geq 0}\frac{2^i}{L}\sum_{j\geq i} c_{l,j}(b)\frac{L}{2^{j-1}} &= g\sum_{j\geq 0}\frac{c_{l,j}(b)}{2^{j-1}}\sum_{i=0}^{j} 2^i \\ &< 4g\sum_{j\geq 0} c_{l,j}(b) \\ &\leq \frac{4g\ t(G,l)}{r+1-\kappa} \qquad (18)\end{aligned}$$

Thus, the expected length of the graph after patching with respect to all vertical quadtree lines (assuming our notion of "vertical" versus "horizontal" is chosen by a preliminary coin flip; in d-space, $d>2$, using a d-faced "coin"), is at most the original length plus

$$\sum_l \frac{4g\,t(G,l)}{r+1-\kappa} \leq \frac{4g\sqrt{d}/d}{r+1-\kappa}\ell(G). \qquad (19)$$

The horizontal lines of the quadtree can be dealt with similarly. But the two sets of lines can *interact* because patches along vertical quadtree square borders can create extra crossings on (necessarily shorter) horizontal quadtree lines, infinitesimally close to their endpoints.

A valid (but weak) response to this quibble would simply be to add at most $2q$ to the lightness r, because if we first patch all the vertical lines, reducing them to r-lightness (but perhaps causing a large number of crossings over horizontal quadtree lines), then patch the horizontal lines, we get a graph G which is r-light for horizontal quadtree lines, $r+2q$-light for vertical quadtree lines, and in expectation at most $(1+\frac{2g\sqrt{2}}{r+1-\kappa})^2$ times longer than G. This proves the first itemized claim of the theorem.

A stronger response is as follows. Regard the quadtree, not as a 2^d-ary tree of cubes, but rather as a *binary* tree (d times deeper) of bricks with longest sidelength : shortest sidelength ratio at most 2 : 1. Then order the ARORA-MODIFY calls along root to leaf paths in this tree. (Notice that we now have a different definition of r-lightness because the tree is different.) In this case, each ARORA-MODIFY call will cause the line (in d-space, hyperplane) it is called upon to become r-light. It may create undesirable extra crossings on other quadtree lines, but only on descendant lines, and never on ancestors. Since these descendants are all going to get repaired by later ARORA-MODIFY calls, this does not matter – we end up with something genuinely r-light.

To bound the expected length increase we may use the following:

Lemma 11 *Let a_1, a_2, ... a_k be real with $0 < a_i$ and $\sum_{i=1}^k a_i = S$. Then*

$$\prod_{i=1}^{k}(1+a_i) \leq (1+S/k)^k < \exp S. \qquad (20)$$

Proof sketch: (20) arises from "Jensen's general argument" (see any book on inequalities) by considering its logarithm and using the fact that $\ln x$ is concave-down. □

This (where $1+a_i$ are the "length amplification factors" caused by the patchings) proves the second itemized claim. □

Remark. The probability that a nonnegative random variable is below twice its expectation is $\geq 1/2$. Hence, the additive increase in length caused by the proof procedure above, will be no more than twice its expected value, at least half the time.

Lemma 12 (Runtime for graph lightening) *After a precomputation consuming $O(N \log N + E_G)$ time (where E_G is the number of edges of G): The proof procedure above can be implemented to run in time linear in the number C of quadtree-G crossings. If we only want r-vapidity rather than r-lightness, the time is linear in the number of relevant crossings.*

Proof. We've already seen that the number of patching operations is linear in C, and clearly each patching operation takes constant time if κ is constant and the locations of the crossings being patched are specified. The only difficulties, then, are finding the crossings in the first place and telling which crossings to patch, each time we patch.

If G obeys the Rounding Assumption, the number of crossings of the spanner over the L possible splitting lines (at integer coordinates) of the quadtree is $O(ds^3 N/\delta)$ by lemma 7. Furthermore, by sorting the endpoints of the spanner edges in each coordinate and doing a 1-dimensional scan on that coordinate, we can find all of these crossings in time $O(N \log N + E_G + C)$ time. Even without the Rounding Assumption the number of relevant crossings is $O(E_G \log N)$ and we may find them in $O(E_G \log N)$ time.

Finally, if every crossing knows which quadtree box it is in and the boxes know their contents, then telling who to patch is immediate. This "knowledge" could be maintained in doubly linked lists precomputed in $O(\log N)$ time per crossing, or only $O(1)$ time per crossing if we have the "floor" function. □

Now, we will discuss spanners, and modifications of the spanner graph.

Lemma 13 Spanner-TST lemma. *A TST exists, which lies inside any $(1+\epsilon)$-spanner graph, and is at most $1+\epsilon$ times longer than the optimal TST. This TST does not use a spanner edge more than twice (i.e. no more than once in each direction).*

Proof: The $1+\epsilon$ approximation is a trivial consequence of spanner-hood. Now suppose (for a contradiction) that the shortest in-spanner TST traverses a spanner edge AB more than 2 times. In that case, the TST (with multiple edge traversals counted multiply) is an Eulerian[15] multigraph. If every k-time multiple edge in this multigraph, $k \geq 3$, is replaced by a j-time multiple edge, with $j \in \{1, 2\}$ and j having the same parity as k, the result is *still* an Eulerian multigraph, and hence has an Euler tour, which would be a shorter in-spanner TST – a contradiction. □

We now provide a patching procedure for spanners that proceeds as follows for a spanner G and a line segment S. One draws two lines along *both* sides of and parallel to S and joins them (crossing S) at a point in the middle. Each edge of G that crosses S is cut at S into two line segments: one to each of the two adjacent lines. This is a patching procedure with $g = 2$, $q = 1$, and $\kappa = 1$. (Really it is going to have $g = 4$, $q = 2$ and $\kappa = 2$, if we consider possible 2-time edge traversals by the tour, see below.) See figure 1.

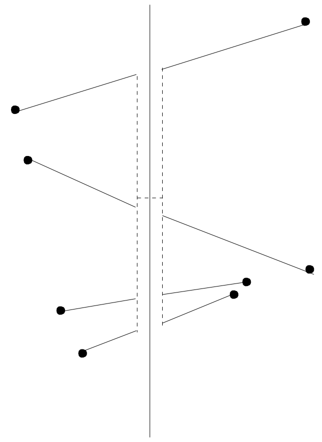

Figure 1: Proof of the patching lemma for spanners.

Lemma 14 (Spanner-TST patching lemma) *Let S be a line segment of length ℓ, and let G be a graph drawn in the Euclidean plane and let π be a cycle in G. If G is patched with respect to the line segment S using the procedure above, then there is a cycle π' that visits all the sites that π visited and whose total length is at most an additive amount that is twice the increase in the cost of G introduced by the patching procedure, i.e., an additive amount of 4ℓ larger than the length of π.*

Proof: If π used an edge that intersected S, consider routing it along the first segment then along one of the parallel lines to the joining point and then to the next segment and finally to the final segment of the edge. This cycle may re-use portions of the parallel lines many times. But we can now modify this cycle to a cycle π' that contains any subinterval of each of the parallel lines at most twice, as in the proof of lemma 13. □

By combining theorem 10 with lemma 14, we can derive the following theorem.

Theorem 15 *Given a traveling salesman problem on a grid, and a quadtree dissection with random shift (a, b), then with probability $1/2$, one can find an $O(s^4)$-light graph G that is* guaranteed *to contain a TST that has cost $1 + 1/s$ times the cost of an optimal TST.*

Proof. Find a $(1+\epsilon)$-spanner with $\epsilon = 1/(2s)$ that has total length $O(\ell(MST)/\epsilon^3)$. Then for a random shift, use theorem 10 to patch it to be r-light. By choosing r to be at least cs^4 for an appropriate constant c, we ensure

[15] connected and with every valency even

that with probability $1/2$, the total increase in length of the spanner will be bounded by $1/(4s)$. If the increase in length is bounded by $1/(4s)$, output the graph. Otherwise fail.

If a graph is produced, it contains a $1 + 1/(2s) + 2/(4s) = 1 + 1/s$ approximate TST by lemmas 13 and 14. □

We note that if a graph is produced it is *guaranteed* to contain an $(1 + 1/s)$-approximate TST. This can be contrasted with Arora's results, where the final output is a $(1 + 1/s)$-approximate TST with probability $1/2$.

2.1 A randomized $(1+\epsilon)$-approximate TST algorithm (when $d = 2$)

As usual in this paper we will set $s = 1/\epsilon$ for convenience.

Algorithm.

1. Scale and pre-perturb the N sites according to the Perturbation Lemma 3 (also removing duplicates) so that they have distinct integer coordinates[16] of size $O(Ns)$.

2. Find a t-spanner graph S of the N sites according to theorem 45. Let this spanner have total length L_S.

3. Find a randomly shifted quadtree in $O(N \log_2(sN))$ time.

4. Modify spanner S according to the Arora modification procedure of theorem 10 so that the resulting graph S' is r-light with respect to the quadtree. Remarks: (i)S' may no longer be a spanner. (ii) This can be accomplished in $O(s^3 N)$ time since the total number of crossings is $O(s^3 N)$ by the Rounding Assumption. See lemma 12. (iii) Even without the Rounding Assumption we would still be able to get an r-*vapid* graph in $O(s^3 N \log N)$ time by lemmas 7, 12.

5. Find the shortest r-light TST which lies inside S', by dynamic programming on the quadtree. (Namely, in each box in the quadtree, we have a TST problem with "boundary conditions" at the r points where the modified spanner crosses the box boundary. There are at most 4 possible ways the tour could cross (or not cross) at such a point, plus there are $2^{O(r)}$ possible planar matchup conditions the tour must obey on each side of the boundary[17] hence at most $2^{O(r)}$ boundary conditions total. We tabulate the min-length solutions to all subproblems for all boundary conditions in all boxes. To combine two boxes to get a min-length subproblem solution in a larger box, we must consider all possible pairs of compatible boundary conditions on the boundary between the sub-boxes. There are $2^{O(r)}$ such pairs.)

[16] The main purpose of the pre-perturbation is to force the quadtree decomposition to stop after $\log_2 N$ levels. Also it keeps our arithmetical precision requirements low – we assume we are in some model of computation in which arithmetic operations on integers of this size may be performed in $O(1)$ time.

[17] Without planarity a bound $r^{O(r)}$ is obvious. Techniques for showing a $2^{O(r)}$ bound with planarity and generation algorithms to achieve such bounds are discussed in [39].

Definition 16 *Let $a_d(\epsilon)$ denote the minimum number X such that a $(1+\epsilon)$-spanner exists (and can be computed efficiently enough for our purposes, say in $t_d(\epsilon)N + dN \log N$ steps, and requiring $s_d(\epsilon)N$ edges at most) for N points in d-space, which is at most X times longer than the SMT. (These functions are bounded in theorem 45, e.g. $a_d(\epsilon) = (d/\epsilon)^{O(d)}$.)*

Let the length of the minimum spanning tree be $\ell(MST)$. Let $t = 1 + \epsilon$, and hence by the Arya et al. spanner construction $\ell(S) \le a_d(\epsilon)\ell(MST)$, and of course $\ell(MST) \le \ell(TST) \le 2\ell(MST)$. (In the present analysis $d = 2$, of course.)

We then claim the TST our algorithm produces will be at most a factor $(1 + O(\epsilon + r^{-1} a_2(\epsilon))$ times longer than the optimum TST. We may choose $r = O(a_2(\epsilon)/\epsilon)$ to get a $1 + \epsilon$ approximation.

The (expected) time to produce the original randomly shifted quadtree is $O(dN \log N)$, i.e. $O(dN)$ time per quadtree level. The space needs are $O(s_d(\epsilon)N)$ to store the spanner. It takes time $t_d(\epsilon)N \log N$.

Finally, performing the dynamic programming requires tabulation of TST subproblems with boundary conditions in each quadtree square, with $\le 2^{O(r)}$ possible kinds of boundary conditions. Two tables corresponding to two subproblems in adjacent quadtree rectangles, may be combined to produce a table of solutions for the combination quadtree rectangle, in time $2^{O(r)}$ by simply considering every possible way to combine two subproblem solutions with compatible boundary conditions. All such combining takes $O(2^{O(r)}N)$ total time – but only $O(2^{O(r)} \log N)$ space is required if we do the dynamic programming in a depth-first order on the quadtree (we only need to store tables associated with the current path down the quadtree, not the whole quadtree).

The algorithm succeeds if the randomly chosen quadtree happened to satisfy theorem 10, which is a probability$\ge 1/2$ event.

Remark. Throughout the argument we have for clarity "cheated" slightly by saying "s^4, s^3, ϵ^3" when in fact the precise values 4 and 3 have nowhere been justified. However, it is justified to say "$O(1)$" instead of these values (see §6).

We conclude

Theorem 17 Main theorem ($d = 2$): *The expected runtime for our randomized algorithm for computing a $(1 + 1/s)$-approximate TST (which succeeds with probability $\ge 1/2$) is $O(2^{s^{O(1)}} N + dN \log N)$. The space requirement is $s^{O(1)}N + 2^{s^{O(1)}} \log N$.*

2.2 Derandomization

Arora's algorithm had involved the use of a random quadtree, for which, he showed, an r-light TST would, with probability $\geq 1/2$, exist, which was only $1+O(1/r)$ times longer than the optimal TST. This had the disadvantage that even if one happened to choose the best quadtree, one had no way of knowing it was the best, or even that it was good enough. Thus, Arora's algorithm outputs a TST, and with probability $\geq 1/2$ ("success") this TST will be as short as Arora's bounds assure; but there is no way to recognize success or failure with certainty.

Arora had little choice but to depend upon randomness, because he did not know the optimal TST.

We, however, do know the spanner graph. This gives us the hope of being able to derandomize the quadtree, in fact the hope that we may specifically choose a quadtree so that a short r-light modified spanner exists, in fact even the hope that we can find the *best* quadtree.

Or instead, we may just choose the quadtree at random just as Arora does, but with the difference that if we succeed, we will know it.

A key fact underlying our derandomization procedure will be that the averaging argument in the proof of theorem 10 did *not* need to average over all L^d possible shifts of the quadtree; instead it only needed to average over Ld shifts, i.e. L in each dimension. We can use a 1D shift optimization procedure to duplicate this, and bound the error we introduce by so doing.

We now give an algorithm for derandomization. Our algorithm will find the approximately best shift for the quadtree with respect to the modification procedure of theorem 10 applied to a spanner.

2.2.1 *Preliminaries*

The number of crossings of the spanner over the L possible splitting lines (at integer coordinates) of the quadtree is $O(ds^3N/\delta)$ by lemma 7. Furthermore, by sorting the endpoints of the spanner edges in each coordinate and doing a 1-dimensional scan on that coordinate, we can find all of these crossings in $O(dN \log N)$ time. We assume now that these crossings are available with each line of the quadtree dissection.

We will use the following fact about binary trees. We first define the level of the root of a binary tree to be 0 and the level of the children of a node to be one higher than its level.

Lemma 18 *Consider an in-order ordering of the nodes of a complete* binary *tree. (Each node is placed between the orderings of the subtrees rooted at it.) For any k successive nodes in the ordering, there is a single node that has lowest level of any of the k nodes.*

Proof: Consider any two nodes at the same level. Consider their least common ancestor. This node is in between the two nodes in an in-order traversal since they are in different subtrees. Moreover, it has lower level than the two nodes. □

2.2.2 *How to find an approximately best horizontal quadtree shift*

Suppose we want to find the best of all the L possible shifts of the quadtree in *one* of the coordinates. By "best," we mean the one causing the least added length when we modify our spanner to make it r-light. (Actually we will be satisfied below if we are off by a constant factor, i.e. are $2r$-light.)

In order to compute the horizontal shift, we assume that the vertical shift is fixed to some arbitrary number b.

Our algorithm will proceed via "dynamic programming" to build a table giving, for sets of $k = 2^p$ successive vertical lines, the cost of applying ARORA-MODIFY to them for each of L possible horizontal shifts.

Each table entry is

- indexed by a minimal level number i, $0 \leq i \leq 1 + \log_2 L$ and a shift $j \in \{0, 1, ..., k-1\}$,
- and contains the cost of applying ARORA-MODIFY to all of these vertical lines assuming that the jth line has level i and no other line has a lower numbered level. (Note that the levels of all the other lines in the set is known given this information by lemma 18.)

We can compute a table entry for $2k$ successive vertical lines from the two previously computed table entries for the left and rigth half-sets (of k lines each), adding costs of the corresponding table entries. This can be done in constant time per entry.

The table for a *single* line l consists of entries for each $j \in \{0, 1, 2, \dots, 1+\log_2 L\}$ that contain the cost of applying ARORA-MODIFY to l assuming that l's level is j. We can compute the table from a single run of ARORA-MODIFY on the vertical line l assuming that $i = 0$. This is due to the fact that ARORA-MODIFY(i, l, b) already computes the cost of ARORA-MODIFY$(i+1, l, b)$. Moreover, we can do this run of Arora's procedure in time proportional to $C \log L$ where C is the total number of crossings of the line l.

We summarize:

Lemma 19 *The horizontal derandomization procedure takes a graph G and a sequence of L vertical lines and finds a value a such that applying ARORA-MODIFY to G to the vertical lines in a quadtree with shift (a, b) produces a graph G' which is r-light with respect to all the vertical segments of the quadtree. Moreover, $\ell(G') \leq 1+O(1/r)\ell(G)$. It requires $O(C \log L)$ time if the graph's edges cross the vertical lines a total of C times.*

We fixed the value of b here and in the ARORA-MODIFY procedure and the final quadtree dissection may use a different value for b. We, however, just run it for this value of b and note that the resulting graph will be $2r$-light with respect to that quadtree with any shift b'. This follows from the fact that any line segment in a quadtree with vertical shift b' overlaps at most 2 line segments for a quadtree with vertical shift b, i.e.:

Lemma 20 *Applying ARORA-MODIFY to produce an r-light graph with respect to the set of vertical lines in a quadtree dissection with shift (a, b) will produce a graph that is $2r$-light with respect to the vertical lines in a quadtree dissection with shift (a, b') for any $b' \in \{-L/2, L/2\}$.*

2.2.3 How to find an approximately best 2-dimensional quadtree shift

Algorithm. First we choose an arbitrary vertical shift b. Assuming this, we use the procedure of lemma 19 to find an approximately best horizontal shift a. We then apply ARORA-MODIFY to the graph and the vertical lines of the quadtree with shift (a, b) to get the modified graph G'. We then use lemma 19 to find an approximately best vertical shift b' for G', now assuming the horizontal shift a is fixed. Finally, by applying ARORA-MODIFY to the horizontal quadtree lines using shift (a, b'), we get our final graph G''.
Analysis. By lemmas 19 and 20, G' will be $2r$-light for the vertical lines (for any horizontal shift b). Also:

Lemma 21 *The number of crossings between edges in G' and the horizontal lines is at most the number of crossings between edges in G and the horizontal lines plus $2q/(r-\kappa)$ times the number of crossings between edges in G and the vertical lines.*

Proof: Each application of a patching procedure on a graph and a vertical line segment introduces at most $2q$ crossings between the resulting graph and a horizontal line segment. The lemma follows by noticing that the total number of applications of a patching lemma during the application of ARORA-MODIFY to all of the vertical lines is bounded by $1/(r-\kappa)$ times the number of crossings between the original graph and the vertical lines. □

The lemma implies that for large enough r the number of crossings of G' with horizontal lines is approximately the same as the number of crossings between G and the horizontal lines.

We claim G'' is r-light with respect to the horizontal line segments of a quadtree with shift (a, b') by analogy with lemma 19.

The final patching to get G'' (which is done with completely known shifts in all dimensions) may have introduced up to $4q$ extra crossings on vertical line segments by an argument similar to the argument at the end of the proof of theorem 10.

By lemma 20, G' was $2r$-light with respect to the vertical line segments in the shifted quadtree. Thus, G'' is $2r+4q$-light with respect to the shifted quadtree. Moreover, the total length of G'' is at most $(1+O(1/r))^2$, i.e. $1+O(1/r)$, times the length of G by lemma 19.

Theorem 22 *Given a 2D graph G, a shift (a, b) can be found deterministically that can be used with the ARORA-MODIFY and a patching procedure to produce an m-light graph G' such that the length of G' is at most $1+O(g/(m+1-\kappa))$ times the length of G.*

The running time is $O(C \log L)$ for the calls of our procedure on single lines and $O(L \log L)$ for all of the other calls. That is (by lemma 7), it is $O(N \log N)$ total for a graph that obeys the Rounding Assumption.

2.3 A faster randomized (but Monte Carlo) algorithm

We can prove the following theorem by combining our structure theorem 10 for spanners with the same theorem but for TSTs:

Theorem 23 *For a quadtree dissection with a random shift, with probability $1/2$, there is a spanner graph G drawn in the plane that contains an $(1+1/s)$-approximate TST tour, π, where*

- *G crosses any square boundary at most $O(s^{O(1)})$ times,*
- *π only crosses any square boundary at most $O(s)$ times.*

With this theorem, we can modify our previous dynamic program to only enumerate tables for subtours that cross out of squares at most $O(s)$ times as a subset of $O(s^{O(1)})$ possible places. Since there are $s^{O(s)}$ such subsets, this leads to an running time bound for the dynamic programming that is $s^{O(s)}N$. This should be compared with our deterministic bound which features behavior like $2^{s^{O(1)}}N$, or to Arora's result which is $(s \log N)^{O(s)}N$.

3 Higher Dimensions

We sketch the differences one needs to extend our approach to higher dimensions.

3.1 Theorems

The Rounding Assumption, Perturbation Lemma 3, "Relevance" definition 6, Lemma 7 bounding quadtree-G crossings, and definition 8 of "r-light" and "r-vapid" already have been stated in a form valid for general d.

The definition 4 of "$t(G, l)$" is readily generalized to "$t(G, H)$" where H is now a $(d-1)$-dimensional box (mutual boundary of two d-boxes) rather than a 1-dimensional line segment (mutual boundary of two squares). Similarly lemma 5 remains valid in d dimensions if we replace "l" with "H" everywhere and sum over all d coordinate directions instead of 2.

We generalize the notion of a patching procedure as follows:

Definition 24 *A (g, κ)-patching procedure for a graph G in $\Re^d$ and a $d-1$ dimensional hypercube H of side-length L forms a graph G' which crosses H only κ times and*

$$\ell(G') \leq \ell(G) + g k^{1-1/(d-1)} L, \tag{21}$$

where k is the number of times edges in G crossed H.

Why are patching procedures going to exist, for graphs such as TST, SMT, and spanners, which meet this definition?

Consider the following patching procedure for spanners in $\Re^d$. Suppose k edges of G that intersect a $d-1$ dimensional hypercube of sidelength L. We assume that no node is contained in the $d-1$ dimensional hypercube (since we could use quadtree planes that are off by an infinitesimal amount from the integral grid containing the nodes). Consider the k points that are intersection points of edges in G with H. We take the MST of these points in H. Then, we patch by making two copies of this MST, one infinitesimally to the left of the plane and one to the right, and adding a single infinitesimal length edge between the two MST copies. (Algorithmically, we may use $(1+\epsilon)$-approximate MSTs as in §1.7.)

We could also define a "(g,κ,q)-patching procedure" by allowing, e.g. at most q traversals of any G-edge (for example, our TST will use a modified spanner edge at most twice).

Lemma 25 *If $d \geq 3$ and $N \geq 2^{d-1}$, then the above is a $(6\sqrt{d},1)$-patching procedure for any graph G in d-space.*

Proof. This follows from lemma 26. □

Lemma 26 (MSTs in cubes are short) *The MST of N sites in (or on the surface of) a unit d-hypercube has length $< \frac{d^{3/2}}{d-1}N^{1-1/d}(1-N^{-1/d})^{-1}$.*

Proof. (Cf. lemma 35.) Rescale the cube to have side $1-N^{-1/d}$. Then by considering their d-volume, we see that it then is impossible for cubes centered at the sites and of side $N^{-1/d}$ all to be disjoint. Hence, at least one site has a neighbor within distance $\sqrt{d}N^{-1/d}$. Draw the edge between these two closest neighbor sites and then mentally remove one of them. Now continue recursively to draw a spanning tree of the remaining $N-1$ sites. The total length of the spanning tree we construct will be $\leq \sqrt{d}\sum_{m=1}^{N} m^{-1/d}$, which is $< \frac{d^{3/2}}{d-1}N^{1-1/d}$. Rescaling, we get the stated bound. □

Lemma 13 is unchanged, and lemma 14 holds trivially.

We now claim that our proof (see also Arora's proof in [1]) of theorem 10 generalizes to $\Re^d$ to yield

Theorem 27 *Given a graph G on a grid, a (g,κ)-patching procedure for the graph and line segments in $\Re^d$, an integer r with $r \geq \kappa$, and a quadtree dissection with a random shift $(a_1,\ldots,a_d)$, then one can use a sequence of patching procedure calls to produce an r-light graph G' that is at most*

$$\exp\left(\frac{4g\sqrt{d}}{r+1-\kappa}[r+1]^{1-1/(d-1)}\right) \tag{22}$$

times longer (in expectation) than G. If d, g, and κ are regarded as fixed while r is allowed to vary, this is just $1+O(r^{-1/(d-1)})$, achieved with probability $\geq 1/2$.

Proof sketch[18]. We'll only outline the differences between the 2D proof (theorem 10) and the present proof valid for general $d \geq 2$.

The ARORA-MODIFY algorithm needs to change by changing "vertical lines l" to "vertical hyperplanes H" and by changing "the segment of l between the y-coordinates ..." to "any quadtree subbox of sidelength $2^{-j}L$." Inequality (16) remains valid except that "$t(G,l)$" changes to "$t(G,H)$." The development up to inequality (19) changes only by the insertion of factors of $C_p^{1-1/(d-1)}$ on various right hand sides, where C_p is the number of crossings being eliminated during the pth call to the patching procedure.

Hence it is important to bound the *average* value of $C_p^{1-1/(d-1)}$, averaged over p and also with a later outer averaging over which vertical quadtree hyperplane H it is. Let C be the total number of crossings of G over H.

Clearly an upper bound on the average value of $C_p^{1-1/(d-1)}$ is C times the average value of $C_p^{-1/(d-1)}$. Now use the fact that $x^{-1/(d-1)}$ is a concave-up and decreasing function of x if $d \geq 2$ to see that our upper bound would be maximized, subject to the constraints that $\sum_p (C_p - \kappa) \leq C$ and $C_p \geq r+1$, if C_p were always identically equal to $r+1$.

The outer average value of C may be bounded by lemma 5, and this, when $d=2$, had already been done when deriving inequality (19).

As a consequence of this, we finally see that the concluding upper bound on $\ell(G')-\ell(G)$ (that is, the right hand side of inequality 19) is still valid in d dimensions if we multiply it by $(r+1)^{1-1/(d-1)}$.

We then continue to follow the proof of theorem 10 using the "stronger response" to the "quibble" and ignoring the weaker one, and noting that we get a factor of d rather than 2 since there are now d coordinates, of which the vertical one is just 1. □

We then get

Theorem 28 *Given a traveling salesman problem on a grid, and a quadtree dissection with random shift $(a_1,\ldots,a_d)$, then with probability $1/2$, one can find an $O(s\sqrt{d})^{d-1})$-light graph G that is* guaranteed *to contain a TST that has cost $1+1/s$ times the cost of an optimal TST.*

The proof of this theorem is analogous to the proof of theorem 15.

3.2 Randomized Algorithms

So, one now can assume that one only needs to find a TST in a modified spanner graph G' where G' is $r = O((s\sqrt{d})^{d-1})$-light.

Optimal TST's can be found in such graphs in $O(r)^r N$ time and space by dynamic programming just as in §2.1 or [1].

[18] Not surprisingly, this generalization of our proof of theorem 10 to general d is similar to Arora [1]'s section 2.3 generalizing *his* proof.

3.3 Derandomization

The derandomization procedure is similar to the procedure of §2.2 but with "line" changed to "hyperplanes" and one needs to process more dimensions. The primary technical difference is in the generalization of lemma 20. For higher dimensions, we use the generalization below.

Lemma 29 *Applying ARORA-MODIFY to produce an r-light graph with respect to the set of vertical lines in a quadtree dissection with shift $(a_1, \ldots, a_d)$ will produce a graph that is $2^d r$-light with respect to the hyperplanes in a quadtree dissection with shift $(a, a'_2, \ldots a'_d)$ for any $\{a'_2, \ldots, a'_d\} \in \{-L/2, L/2\}^{d-1}$.*

With this lemma, the derandomization procedure and its analysis proceeds in an analogous manner with that of section 2.2.

3.4 Faster Randomized (but Monte Carlo) Algorithms

Section 2.3 developed a Monte Carlo algorithm for finding TST's that is based on combining our structure theorem for spanners and (Arora's) structure theorem for TST's in the plane. Since analogous version of both theorems apply in higher dimensions, one can develop a Monte Carlo algorithm for higher dimensions whose behavior with respect to s is no worse than Arora's while maintaining our improved dependence on N. In this case, however, the differences in running time are perhaps less clear.

4 Extension to other graphs besides TST

Our spanner-based TST methods do not extend immediately to SMTs because SMTs include extra (Steiner) points as vertices, and these extra vertices are not vertices of a spanner!

They also don't extend immediately to such graphs as MM, M2M, and EC, because these graphs can be hugely (i.e. more than a constant factor) shorter than the MST and hence any spanner. Thus step 4 of our TST algorithm, which modifies the spanner graph to make it "r-vapid," could immediately kill us by introducing extra length which, while bounded by a small constant times the length of the spanner, still is hugely greater than the length of the graph we are trying to approximate! Another problem is the "rounding coordinates to integers" pre-perturbation step 1 of the TST algorithm, which was needed to keep the quadtree $O(\log N)$ levels deep. Because MM could (say) have total length $N^{-17} \cdot \ell(MST)$, rounding all coordinates to make the sites lie on an $O(N^2) \times O(N^2)$ grid would not be wise.

However, we will now demonstrate how to get around these obstacles in the cases of SMT, RSMT, EC, and M2M. (We still don't know how to avoid them and get a $(1+\epsilon)$ approximation algorithm for MM. However, in §4.4 we'll show how to get a Monte Carlo algorithm to approximate MM to within a factor of $2^{O(\sqrt{d})}$ in $O(N \log N)$ time.)

For SMT, the idea is to replace the spanner graph on our point set, by a "banyan." This required discovering several new facts about SMTs (of independent interest) and indeed the discovery of banyans themselves.

For EC and M2M, we replace the spanner graph by a different graph G which is only a constant factor longer than EC or M2M but still has the distance approximation and patching properties we need. Our usual algorithmic machinery then grinds on merrily but using this graph instead. Rounding difficulties are avoided by treating each connected component C of G as an independent subproblem. Since (as we will show) the part of EC or M2M whose vertices lie in C has length of the same order as C itself, the usual rounding procedures applied to each subproblem will suffice.

For MM, the idea is to avoid modifying the spanner to make it r-light. Then we find the best 1-light or 1-vapid OVS (a graph related to MM) inside the spanner by dynamic programming – which we can still do efficiently, because the number of subproblems arising in the dynamic programming is only $O(N)$ due to *its* 1-lightness. This idea can be applied, more generally, to find the best 1-light subgraphs of all kinds inside a spanner; and thanks to the fact that our version (theorem 10) of Arora's structure theorem allows 1-lightness, it will yield constant factor approximation in fixed dimension.

4.1 Banyans

$O(N \log N)$ algorithm for constructing a $(1+\epsilon)$-banyan of N sites in d-space. (See figure 2.)

1. Find a $(1+O(\epsilon))$-spanner graph as in §6.
2. Circumscribe a d-sphere about each spanner edge.
3. Expand all these d-spheres about their centers by a factor κ_d/ϵ (κ_d is a constant depending only on d). We now have $(d/\epsilon)^{O(d)} N$ d-spheres.
4. In each such d-sphere of radius r, place a grid of new points, with grid spacing $\kappa_d^{-2} \epsilon^2 r$. This yields a point set with $\kappa_d^{2d} (d/\epsilon)^{O(d)} N$ "Steiner candidate" points.
5. Finally, output a $(1+O(\epsilon))$ spanner of both the original sites and the Steiner candidate points.

Theorem 30 (Banyan theorem) *Let $0 < \epsilon < 1$. If κ_d is a sufficiently large (it will suffice if $\kappa_d = d^{O(d)}$ and if $\kappa_2 = 303$) then the graph described above is a $(1+\epsilon)$-banyan. If d and ϵ are fixed: The algorithm described above runs*[19] *in $O(N \log N)$ time and consumes $O(N)$ space. And if the spanners used in the algorithm have bounded valencies and are at most a constant factor longer than the SMT of their vertices, then the banyan*

[19]More precisely the time is $(d/\epsilon)^{O(d)} d^{O(d^2)} N + O(dN \log N)$ and the space is $(d/\epsilon)^{O(d)} d^{O(d^2)} N$. With $\kappa_d = d^{O(d)}$ the banyan is $d^{O(d^2)} \epsilon^{-O(d)}$ times longer than the SMT.

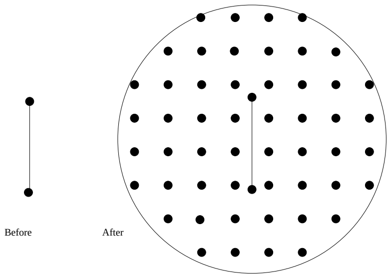

Figure 2: Conversion of a spanner edge into a set of Banyan points. The circle has radius $\kappa_d L/\epsilon$ where L is the length of the edge. The grid has spacing $\epsilon L/\kappa_d$.

produced will also have bounded valencies and will be at most a constant factor longer than the SMT of the sites.

Theorem 30 will be proven in 2D in the next subsection, and in full generality in the one after. (For properties of spanner graphs and runtime analysis for spanner finding algorithms, see §6.)

Remarks.

(i) It is also possible to force the banyan to have small graphical diameter. See [2].

(ii) One reason it is amazing that a $(1+\epsilon)$-banyan requires only a linear number of Steiner points if $\epsilon > 0$, is that a 1-banyan apparently could require as many as $N2^N$ of them for the "ladder" 2D site set: a $2 \times (N+1)$ grid. (A 1-banyan necessarily includes the union of the Steiner minimum trees for all the site subsets.)

4.2 Steiner minimal trees (SMT and RSMT) in 2D

Assuming one accepts theorem 30, the rest is easy.

Corollary 31 *Selecting the Steiner points of an SMT of (any subset of) our sites solely from our $O(N)$ "Steiner candidates" (and the edges of our SMT purely from the banyan edges) will cause the SMT length to increase at most by a fraction $(1+\epsilon)$.*

And our usual algorithmic approaches starting with the banyan instead of a spanner will generate an $O(1+\epsilon)$-optimal 2D SMT. (We leave the precise formulation of the appropriate patching lemmas and subproblem definitions in the dynamic programming, to the reader.) Admittedly there will be a extra factor of $\epsilon^{-O(1)}$ in our time and space bounds as compared to our algorithm for 2D TST, but that is all.

In the discussion below, we will use "sites" to denote original sites, i.e. the input of the problem; and "Steiner points" (we will always use "points" not "sites") to denote extra vertices of the SMT which are not inputs. This convention will be so important for the reader's comprehension that we enshrine it as a "definition:"

Definition 32 *"Steiner points" are vertices of the SMT that are not members of the original set of N "sites."*

Proof of theorem 30 in 2D. We first remark that we don't need to prove theorem 30 for *any* site subset, merely the whole set, because it is obvious that, e.g., a t-spanner for the whole set is a t-spanner for any subset, so our construction algorithm obviously produces something that has the same properties for any subset of the sites, as it has for the whole set.

The proof will ultimately turn on a fairly complicated case analysis. But before we can reach that point, we will prepare the ground by an escalating sequence of subclaims. We begin with some well known and easy facts about SMTs.

Fact 0: SMT is a tree. In any metric space, its length $\ell(SMT)$ obeys $\frac{1}{2}\ell(MST) < \ell(SMT) \le \ell(MST)$. The length of the SMT of any site subset does not exceed the length of the SMT of the full set[20].

Fact 1: Every steiner point s of SMT has valence 3 and the 3 SMT edges meeting s lie in a 2D plane (no matter how many dimensions we are in) and have mutual angles 120°.

Fact 2: If there are N sites, then there are at most $N-2$ Steiner points in the SMT and at most $2N-3$ SMT edges.

Fact 3: No two coterminous edges of SMT meet at an angle of $< 120°$. Every SMT vertex has valence≤ 3.

Fact 4: Two edges of SMT intersect only at a common endpoint.

Fact 5: (Empty lune property; also true for MST) If AB is an SMT edge, then the "lune" of that edge is the intersection of the two open balls of radius AB with centers A and B. This lune does not contain any portion of any edge of the SMT (except for the edge AB of course).

Fact 6: (120° Wedge property; only valid when $d=2$) If s is a Steiner point of the SMT then in any closed 120° wedge with apex s, there exists a site v, and there is an SMT path sv which lies entirely inside the wedge.

Fact 7: (MST property) If AB is an SMT edge where A and B are sites, then AB is an MST edge.

The above facts 0-7 were shown in [20]. All of them are valid in general dimension except for fact 6.

Lemma 33 (Hexagon property; $d=2$ only) *Let AB be an SMT edge. Suppose that if you walk along a path of SMT edges starting from A (or from B) for at most 3 edges or until you encounter a site (whichever comes first), then each SMT edge you encounter has length $\le L$. Then: the closed regular hexagon H_A of side L (and hence diameter $2L$) which has a corner at A, and which does not contain the line segment AB, but, if AB were extended into an infinite line, such that this line would split H_A exactly in half, contains a site V_A attached to A by an SMT path lying entirely inside H_A (this site could be A itself). Similarly H_B contains a regular site V_B.*

Proof of Hexagon property: It is almost the same as the proof of lemma 3 (page 841) of [19], but we include

[20] MST does not have this property.

it here for completeness. Refer to figure 3 in which AB and all the hexagon sides are $\leq L$ in length and AB is vertical. Then a 120° wedge (as shown, dashed) must exist containing a site V_A (by fact 6) connected by an SMT path to its apex; and V_A must lie inside hexagon H_A due to fact 4. □

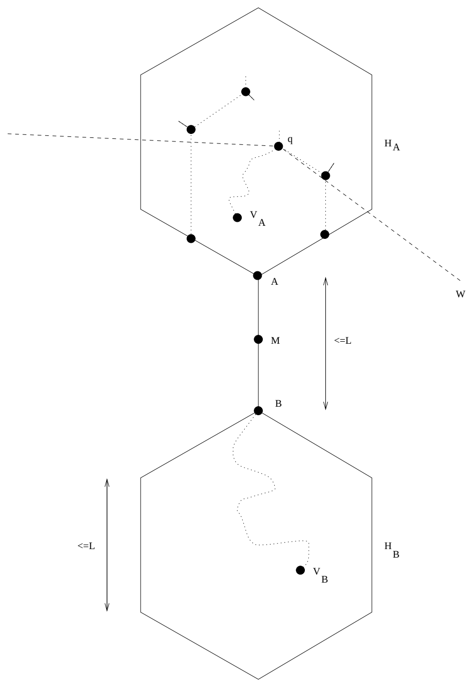

Figure 3: Proof of the hexagon property. 120° wedge W must include a site connected to q by a path contained entirely inside it. All SMT edges shown are $\leq L$ long; the regular hexagon sides are L.

Lemma 34 (Spanner path property) *Let X and Y be two sites. Suppose that if some SMT edge, of length l, were removed from the SMT, then X and Y would be in different connected components of the resulting graph. Then: it is impossible for a path from X to Y, consisting entirely of intersite line segments shorter than l, to exist.*

Proof. Removing the SMT edge and replacing it with some line segment in the XY path, would yield a connecting network shorter than the SMT, contradicting its minimality. □

We now proceed with the proof of theorem 30 in 2D with $\kappa_2 = 303$.

Suppose AB is an SMT edge with length L, and let M denote the midpoint of this edge.

If AB is an MST edge, i.e., if both A and B are sites, theorem 30 is proven since AB must be in the spanner (or a path closely approximating it). So assume A is Steiner.

If some SMT edge attached to A via a path ≤ 3 SMT edges long is longer than $50L/\epsilon$ then it does not matter where A is. More precisely, we need only worry about the long edge and not about the short edges such as AB because we could just collapse AB (or replace it by part of some MST) and this won't make any difference to the approximation properties of our SMT. And so on recursively in fact – if the new much longer SMT edge, at most 3 edges away, that we have just mentally "jumped" to has a much longer (still) SMT edge near it, we continue jumping until we reach an edge e *without* a hugely longer SMT edge at most 3 edges away, and then all we need do is get its two endpoints placed accurately (to within $O(\epsilon\ell(e))$) to get a good SMT. We need only analyze such "locally long" SMT edges because the other SMT edges can be ignored and the cost of ignoring them amortized into the analysis of their nearby locally long edge.

So assume all the edges attached to A via a path ≤ 3 SMT edges long are shorter than $50L/\epsilon$.

Then by the hexagon property, there must exist two sites V_A and V_B in the two regular hexagons H_A, H_B respectively of side $50L/\epsilon$. These two sites are separated by distance S where $L < S \leq L + 200L/\epsilon$.

Now consider the sites, and the edges of the SMT, and spanner, lying within a ball Q of diameter

$$D = (1 + \epsilon/2)S = 101L + 200L/\epsilon + \epsilon L/2. \quad (23)$$

centered at M. (See figure 4.) Ignore everything outside Q.

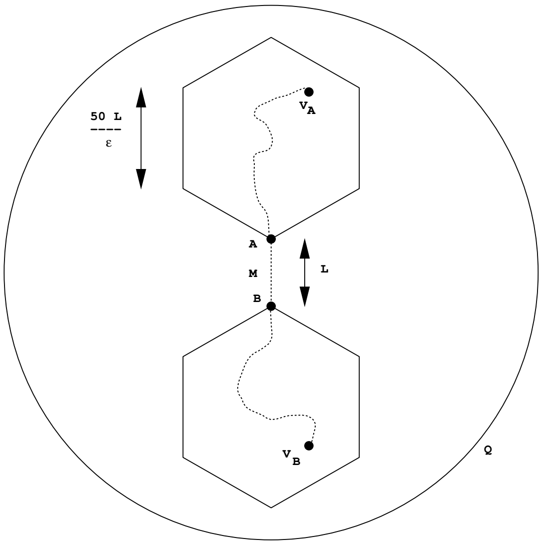

Figure 4: Proof of banyan construction's approximation property in 2D.

By lemma 34 V_A and V_B are not interconnected by a path consisting entirely of "tiny" (that is, length $< L$) spanner edges all lying inside Q.

Hence case 1 or 2 below must hold.

Case 1: V_A and V_B are connected by a spanner path containing a big (i.e. length$\geq L$) spanner edge somewhere in Q. In this case theorem 30 is proven because a sphere circumscribed about this edge and then expanded by a factor $D/L + 1$ contains A and B, hence the grid points in our Banyan construction algorithm arising from this spanner edge (with grid spacing $\epsilon/(D/L + 1)$; or indeed even using spacing 1.4ϵ would suffice, even less if the grid were triangular) will suffice to approximate A and B.

Case 2. They are not connected at all. In this case, our "spanner" could not really be a spanner because the spanner path from A to B would have to go outside Q at some point, causing this path to be too long for the spanner distance-approximation property to hold. This would establish the theorem by contradiction.

Theorem 30 is now established in 2D with $\kappa_2 = (D/L+1)\epsilon = 200 + 102\epsilon + \epsilon^2/2$, i.e., if $\epsilon \le 1$ then $\kappa_2 = 302.5$ suffices, and for all sufficiently small ϵ it suffices if $\kappa_2 = 201$. □

4.2.1 RSMTs

To extend our 2D SMT result to work for RSMT in 2D, we need an analog of the hexagon property (lemma 33); and that requires also an analog of the 120° wedge property. If instead of the 120° wedge property we merely had a 179° property (any constant angle below 180° would do), that would still suffice. But we don't have even this. However, for sites in general position the RSMT has all Steiner valencies 3 or 4, and every angle at a Steiner point (or a site with valency≥ 2) is $< 180°$. Arbitrarily close approaches to 180° can occur, indeed exactly 180° is possible for points not in general position. We therefore, for $(1+\epsilon)$ approximation, restrict our attention to RSMTs drawn with sloped lines (but their length is measured in the L_1 metric) with all angles $< 180 - \epsilon°$. In that case we get a $180 - \epsilon°$ wedge property and a rather giant analog of the hexagon property (we lose another factor or two of ϵ here!) and the rest of the argument follows, although perhaps one or two factors of ϵ weaker.

4.2.2 SMTs in higher dimensions than 2

The key ingredient in our 2D proof of theorem 30 was the hexagon property, lemma 33. Unfortunately, our proof of the hexagon property is inherently 2-dimensional, depending not only on the 120° wedge property, but also on the "hemming in" quality of fact 4 in the plane.

After much effort we were indeed able to generalize the hexagon property to allow $d \ge 3$, obtaining a very complicated proof of a very weak version of the banyan theorem, but one which proved a lot of independently interesting properties of SMTs along the way. But then later we were able to simplify and strengthen the proof drastically – presented below. Unfortunately the simplified proof no longer needs the independently interesting SMT properties, including the generalized hexagon property[21]. We first need the following two lemmas.

Lemma 35 (MSTs in balls are short) *The MST of N sites in (or on the surface of) a unit ball in d-space has length $< \frac{2d}{d-1}N^{1-1/d}(1-N^{-1/d})^{-1}$. If $N \ge 2^d$ and $d \ge 2$, this is $< 8N^{1-1/d}$.*

[21] To prevent it from being totally lost, we mention: **Claim:** If there is a d-sphere with an SMT edge e_1 piercing its surface radially then either the sphere (i) contains a site, (ii) contains $> p$ Steiner points, (iii) the SMT contains a path, whose distance to the sphere center is monotonically decreasing, from e_1 to a point at distance $< \sqrt{1-4^{-p}}$ radii from the sphere center.

Proof. (Cf. lemma 26.) Rescale the unit ball to have radius $1 - N^{-1/d}$. By considering their d-volume, we see that it is impossible for balls centered at the sites and of radius $N^{-1/d}$ all to be disjoint. Hence, at least one site has a neighbor within distance $2N^{-1/d}$. Draw the edge between these two closest neighbor sites and then mentally remove one of them. Now continue recursively to draw a spanning tree of the remaining $N-1$ sites. The total length of the spanning tree we construct will be $\le 2\sum_{m=1}^{N} m^{-1/d}$, which is $< \frac{2d}{d-1}N^{1-1/d}$. Rescaling, we get the stated bound. □

Lemma 36 (Empty ball lemma) *Let there be two concentric d-balls, the outer one B_o of radius 1 and the inner one B_i of radius r, $0 < r < 1$. For any SMT: if B_o is empty of sites, then B_i contains $s = d^{O(d)}$ Steiner points, where the "O" applies in the regime where r is fixed[22] and $d \ge 2$ is unbounded.*

More is true. Let there be an SMT Z of some sites. The number of Steiner points s of any connected component Y of $Z \cap B_i$ obeys $s = d^{O(d)}$ if r is fixed unless *B_o contains at least $d^{\Omega(d)}$ sites, each connected to Y by disjoint SMT paths of length $d^{-\Omega(d)}(1-r)$ and lying entirely inside B_o. [Also, if d is fixed while $r \to 1-$, then $s = 2^{2^{O(1/(1-r))}}$.]*

Proof. Suppose B_o contains M sites, $M = d^{o(d)}$. If $s \le 6M$, we are done, so suppose not. Consider the portion T of the SMT contained in B_o and connected (by a path lying wholly within B_o) to Y.

By facts 1 and 3, T has $N \ge s - 3M + 2$ "leaves," i.e. places where it touches ∂B_o. Hence its length is at least $(1-r)(s+2-3M)$, which exceeds $(1-r)s/2$. Then $(1-r)s < 16N^{1-1/d}$ since the SMT of these $N+M$ sites, $M < N$, has length $< 16N^{1-1/d}$ by lemma 35, assuming $N + M \ge 2^d$. (If $N + M < 2^d$, we are done.) Choose $\delta > 0$. If we make a sequence of $d \log\log N$ concentric balls with gaps of size $d^{-1}\delta/\log\log N$ between them, then we see by applying the preceding inequality one time at each layer, that our innermost ball will have radius $1-\delta$ and it will have at most order $(\delta^{-1} d \log\log N)^d$ points at which T crosses its surface. In fact, choose $\delta = 1/\log\log N$. Then the number of crossings of the surface of the ball of radius $1 - 1/\log\log N$ will be $O(d^d(\log\log N)^{2d})$. Now, repeat *this* argument order $\log^* N$ times, which will suffice to get N down to a constant (more precisely, $d^{O(d)}$). The final result is that in the ball of radius $q = 1 - O(\log^* N/\log\log N)$, there are at most $d^{O(d)}$ Steiner points!

Now to prove the sentence enclosed in square brackets (which is so enclosed because we will never use it): If we now allow r to vary, there are two cases. If $r < q$,

[22] When $d = 2$ and r, $0 < r < 1$, is allowed to vary, it is easy to see that $s + 2 < 2\pi/(1-r)$ since the SMT of $s+2$ sites on a unit circle has length $< 2\pi$. On the other hand, Marcus Brazil showed us a construction in which $s > |\log_{10}(1-r)|$. Brazil constructed an SMT in the plane for N sites, none of which were in a regular hexagon H, but having $N-2$ Steiner points, all of which were in H; and a modification of it also works if H is a circle instead of a regular hexagon.

which will happen if $N > 2^{2^{O(1/(1-r))}}$, the result follows. Otherwise, we may just use $s \leq N-2$. □

Proof of theorem 30 for all $d \geq 2$:

Let the SMT consist of edge AB, of length L, and subtrees SMT_A and SMT_B into which it would be split by the removal of AB. SMT_A of course is the SMT of A and all the sites in SMT_A and hence any theorems about SMTs generally may be applied to it, since it is an SMT; similarly for SMT_B.

Draw three concentric balls B_1, B_2, and B_2 around M, the midpoint of SMT edge AB, of respective radii $1R$, $2R$, and $3R$.

It will suffice (except for problems arising from "overcharging;" see the end of the proof) if $R = \Omega(d \log d)L/\epsilon$.

Explore SMT_A outward from A up to k edges out, $k = \Omega(d \log d)$; denote the resulting portion of SMT_A by "E_A."

We may assume E_A lies entirely inside B_2 since otherwise a "long edge," of length $\geq 2R/k$, must exist in it, and we are done since we can amortize the approximation error of ignoring AB by charging it to the long edge (The overcharging factor is bounded by $\leq 2^k = d^{O(d)}$).

Case 1. E_A has no sites. If so, then it must contain a "long edge," of length $\geq R/k$, in order for it to get out of B_1. It must get out of the B_1 since B_1 contains at most $d^{O(d)}$ Steiner points by the empty ball lemma applied to B_1 and B_2, and $2^k > d^{O(d)}$, – unless B_2 is contains some site connected to E_A by an SMT_A path, see case 2. Hence we can again amortize the approximation error of ignoring AB by charging it to the long edge.

Case 2. SMT_A contains a site lying inside B_2. If so, then (applying the same argument to B) we conclude that there must exist sites V_A and V_B, both inside B_2, in SMT_A and SMT_B respectively.

Case 2a. Then if $\mathrm{dist}(V_A, V_B) < L$, we get a contradiction by lemma 34. Indeed if there is a spanner path from V_A to V_B consisting entirely of edges shorter than L, lemma 34 yields a contradiction.

Case 2b. Hence the shortest spanner path from V_A to V_B must either contain an edge longer than L, or

Case 2c. It must go outside B_3.

In case 2b, we are done because the banyan points arising from the longer than L spanner edges in B_3 suffice to approximate the locations of A and B.

In case 2c, we obtain a contradiction with the spanner distance approximation property for the path $V_A V_B$.

Unfortunately we lost a factor of order 2^k in our "charging" arguments, which we need to compensate for by making R a factor 2^k larger (equivalently ϵ a factor 2^k smaller, or equivalently using $\kappa_d = d^{O(d)}$) which we've done in the theorem statement. □

The result is:

Theorem 37 *We have an algorithm for finding* $(1 + 1/s)$*-approximate SMTs and RSMTs for* N *sites in* d*-space which runs in the same amount of time and space as our* $(1+1/s)$*-approximate TST algorithm on* $s^{2d} d^{O(d^2)} N$ *sites. This is* $O(N \log N)$ *time and* $O(N)$ *space if* d *and* s *are held fixed.*

Remark. The proof above is readily generalized to hold for RSMTs instead of SMTs (e.g. prove a lemma saying RSMTs in L_1 balls are short, etc.).

Remark. Notice that our banyan construction has $d^{O(d^2)} N$ edges, as opposed to spanners, which require only $d^{O(d)} N$. We do not know if this increase is necessary. Notice that this does not hurt our SMT and RSMT time and space bounds because they already depended doubly exponentially on d, compared to which a factor $d^{O(d^2)}$ is insignificant.

4.3 Minimum edge cover (EC) and 2-matching (M2M)

The LP dual [8] of the LP formulation of the M2M problem (cf. [30]), is the problem of finding a non-negative real weight w_v associated with every site v, such that $w_a + w_b \leq \mathrm{dist}(a, b)$ when $a \neq b$, and maximizing $W = \sum_v w_v$.

At optimality, W will be the total length of M2M and the inequalities on $w_a + w_b$ will be tight exactly when ab is an M2M edge.

We may interpret this geometrically as follows. Regard w_a as the radius of a ball centered at a. The inequalities on $w_a + w_b$ cause all these balls to be interior-disjoint. Two balls are tangent iff their centers are joined by an M2M edge.

Now clearly the largest a ball could possibly be is the distance to its center's nearest neighbor. Hence

Lemma 38 *SOI is a supergraph of M2M.*

The SOI graph is computable in $O(d^{O(d)} N \log N)$ time (§1.1). Now consider the connected components of the SOI graph, and for each, consider the MST of its vertices.

Lemma 39 *The total length* $\ell(F)$ *of the resulting forest* F *obeys*

$$\ell(M2M) \leq 2\ell(F) \leq 2^{O(d)} \ell(M2M). \qquad (24)$$

Proof. Draw a cycle traversing each each of each tree in the forest twice, and this is a perhaps non-minimal 2-matching. Hence $\ell M2M \leq 2\ell(F)$. On the other hand, no edge ab in F is longer than the distance from a to its nearest neighbor, plus the distance from b to its, and clearly every edge in M2M must be at least as long as the distance from either endpoint to its nearest neighbor. Since the valency of any site in an MST is $2^{O(d)}$, the result follows. □

So, since the $(1+1/s)$-spanners of Arya et al. (see §6) are $\leq (sd)^{O(d)}$ times longer than the MST of their vertices, we conclude that we can use the union of the spanners of the vertex sets of the connected components of the SOI graph in our TST algorithm, instead of the full spanner graph of all the sites.

Similarly,

Lemma 40 *EC is a subgraph of the "ANN×2" graph in which every site* x *is joined by an edge to each site within distance 2 times the distance to* x*'s nearest neighbor.*

Proof. If some edge ab of EC were over twice as long as the either a or b's nearest neighbor distance, then get rid of it and replace it with the two nearest neighbor distances. □

The ANN×2 graph has $2^{O(d)}N$ edges[23] and is computable in $d^{O(d)}N\log N$ steps using a variant of Vaidya's all nearest neighbor algorithm [43]. Again we may use spanners of its connected components.

In any metric space the EC graph consists of a union of disjoint single edges and "L's" (2-edge paths). (Proof: if there were a (≥ 3)-edge path in EC, you could remove the middle edge. If there were a (≥ 3)-valent vertex you could remove 2 of its edges and join the 2 neighbors directly.) Also in any metric space, a version of EC, call it "EC*," permitting Steiner points (that is, EC* is the shortest forest in which every site has valency≥ 1 and every Steiner point has valency ≥ 2) is a union of disjoint single edges, 2-edge paths, and 3-edge stars whose central 3-valent vertex is Steiner. So now obviously, both EC and EC* are approximable to within $1+\epsilon$ using our methods. (Because of its simple structure, it isn't necessary to use banyans to approximate EC*. That would be "overkill." Merely altering the subproblem combining algorithm in the quadtree dynamic programming should suffice.)

Incidentally, our TST algorithm actually gets slightly faster when we turn it into a EC or M2M algorithm, because the number of possible "boundary conditions" on subproblems with r points where we are allowed to cross the boundary, becomes $2^{O(r)}$ instead of $r^{O(r)}$.

Theorem 41 *We have algorithms for finding $(1+1/s)$-approximate EC, EC*, and M2M for N sites in d-space which run in Monte Carlo time*

$$2^{O(d(\sqrt{d}s)^{d-1})}N + O(dN\log N) \tag{25}$$

and space

$$(sd)^{O(d)}N + 2^{O(d(\sqrt{d}s)^{d-1})}\log N \tag{26}$$

or Las Vegas time

$$(sd)^{O(sd)}N + O(dN\log N), \tag{27}$$

or deterministic time (8) and space (9).

4.4 An $O(N\log N)$-time Monte Carlo algorithm to find an $\exp O(d^{1/2})$-approximate Minimum Matching

4.4.1 Even Forest heuristic

We will first show how to produce an $\approx N/2$-ratio approximation to the minimum matching in $O(N\log N)$ (deterministic) time.

The "even forest" heuristic for approximate minimum matchings in any metric space

1. Find the MST of your N sites.
2. Call an MST edge "odd-odd" if its removal would divide the MST into an A-node tree and a B-node tree with A and B both odd. Get rid of all the MST edges except for the odd-odd ones.
3. The result is a forest "EF." Every tree in EF has an even number of nodes.
4. For each tree in EF, find a tour that goes "around" the tree and then pick the smaller of the two matchings inside the tour, and "shortcut" it by straightening bent paths. This is a matching whose length is shorter than the length of EF. (Possible modification: if the tree is of small enough cardinality, say ≤ 8, use the exact min matching of its vertices.)

Lemma 42 *Every tree in EF has an even number of nodes and every vertex in EF has an odd valence. EF is the unique odd-valent subgraph of MST. EF is also the unique minimal length and minimal edge-count forest in which all trees are even.*

Proof. (1) All valencies odd implies number of vertices even since 2 times the number of edges is sum of valencies in any graph – so the sum must have an even number of terms. (2) All trees are even is proven recursively or inductively. (3) All valencies are odd is implied by all tree cardinalities are even because if an even valence existed at some vertex v then the tree that it was in, would have to contain as subtrees when v removed, at least one whose cardinality was even. But then that edge would have been zapped. (4) Uniqueness is because no odd-odd edge may be removed from MST otherwise an odd tree would be forced. Now if any even-even edge were kept, that would cause non-minimality; also it would cause some valency to become even. □

Theorem 43

$$\ell(MM) \leq \ell(EF) \leq \frac{N}{2}\ell(MM). \tag{28}$$

The right hand inequality is valid with any set of positive distances. The left hand one is valid in any metric space (i.e. in which the distances satisfy the triangle inequality so that "shortcutting" makes sense). The upper bound may be approached arbitrarily closely even in the Euclidean plane.

Proof. Clearly $\ell(MM) \leq \ell(EF)$ by step 4 of the algorithm which converts EF into a (perhaps non-optimal) matching shorter than it.

The fact that EF is at most $N/2$ times longer than MM is proven as follows. Minimum weight matching is a constrained minimization problem, with one of the constraints being that at least one edge of the matching must cross every "odd cut" C. (That is, C is the edges ij for $i \in S$ and $j \in \overline{S}$ where $|S|$ is odd.) If we only consider a subset of the constraints, $\ell(MM)$ does not increase. So just consider the odd cuts corresponding to the odd-odd edges of the MST and the site-subsets

[23] Same proof as in footnote 10 for the SOI graph.

they induce by their removal from MST. Since each MST edge is the shortest edge crossing its cut (easily proven by contradiction), and it is impossible for an edge to be "re-used" more than $N/2$ times (i.e. a single edge crossing K cuts would be a re-use factor of K) because the maximum number of odd-odd edges on a single path in the MST is $N/2$ (as may be proven by induction on N), $\ell(EF) \leq \frac{N}{2}\ell(MM)$ follows.

Finally, to see that ratios arbitrarily close to $N/2$ are achievable, consider N points (N even) placed on $N/2$ of the edges of a regular $(N/2+1)$-gon, with 2 points per edge: 1 near each endpoint. In that case the MST is just a path along the $(N/2+1)$-gon boundary which has a gap at its unrepresented edge. EF consists of $N/2$ line segments each only slightly shorter than a gon-edge. This would be the matching the EF heuristic would return. However, the optimal matching is almost $N/2$ times shorter and consists of one longer line segment on the unrepresented gon-edge, plus $N/2-1$ tiny line segments near the gon-vertices. (See figure 5.)□

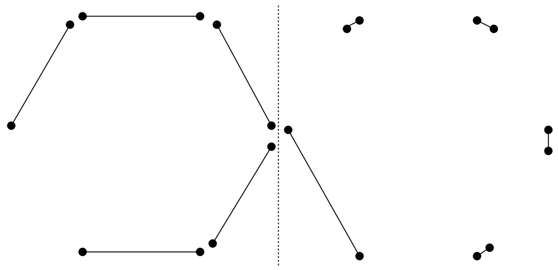

Figure 5: Bad case for "EF" matching heuristic. The matching on the left is produced by the heuristic; the one on the right is optimal.

Notice that we may obtain a $(1+\epsilon)N/2$-approximation of MM in $d^{O(d)}N\log N$ time and $d^{O(d)}N$ space by use of the EF algorithm in the metric space of path-distances in a $(1+\epsilon)$-spanner.

4.4.2 Monte Carlo algorithm

Algorithm

1. Find a 1.5-spanner S in $d^{O(d)}N\log N$ time and $d^{O(d)}N$ space.

2. Find the MST inside S in $d^{O(d)}N$ time.

3. Find an $0.75N$-approximation to MM by applying the EF heuristic (§4.4.1).

4. Let A be a number such that $\ell(MM) \leq A \leq 0.75N\ell(MM)$.

5. Get rid of all the spanner edges which are longer than A.

6. Regard all the nodes from the connected components S_i of the resulting subgraph of S as independent matching subproblems. For each such subproblem i

   (a) Let the total length of the EF$\cap S_i$ be L_i. Perturb all S_i's nodes to lie on a grid with spacing L_i/N.

   (b) Find a 1.5-spanner of these perturbed nodes, and a shifted quadtree, and find the shortest 1-vapid odd-valent subgraph OVS of the spanner by dynamic programming.

7. Convert the odd-valent subgraphs into matchings of shorter or the same length by the usual method of choosing the shorter of the two matchings inside a tour round an OVS.

Theorem 44 *The algorithm above will run in $d^{O(d)}N + O(dN\log N)$ time and consume $d^{O(d)}N$ space, and it will produce a matching whose expected length is at most $1.5\exp(8\cdot 2^{1-1/(d-1)}\sqrt{d})$ times optimal.*

Remark. Unlike our TST algorithm, we do *not* modify our spanner to make it r-vapid. Instead we just find an r-vapid OVS inside the spanner, and we only allow $r=1$. This makes the algorithm fast, but also sacrifices any chance of getting arbitrarily good approximation.

Proof. In the dynamic programming, suppose at some stage that C "relevant" (definition 6) spanner edges cross the quadtree boundary. In that case we must combine $O(C)$ subproblem solutions on the left with $O(C)$ on the right – which may be accomplished in $O(C)$ time since (by the definition 8 of "1-vapid") a problem on the left is compatible with one on the right only if both subproblems use the same boundary crossing spanner edge – which is the case we must worry about if the subproblems both have odd cardinality – or both use none – independent even cardinality subproblems. The total runtime is thus the total number of spanner-quadtree relevant crossings, which is $d^{O(d)}N\log N$ since there are $d^{O(d)}N$ spanner edges, each of which is relevant in $O(\log N)$ quadtree boxes (cf. lemmas 7, 12).

It follows from the structure theorems 10 and 27 (with $g=2$, $\kappa=1$, $r=1$) that the best 1-vapid matching is $\leq \exp(8\cdot 2^{1-1/(d-1)}\sqrt{d})$ times longer than optimal, in expectation with a random quadtree. We sacrifice an additional factor of 1.5 if we use a 1.5-spanner. □

Remark. We would also sacrifice an additional factor of 2 if we want probability of "success" (i.e., getting below twice the expectation value) $\geq 1/2$. But the factor of $1.5\times 2=3$ may be reduced arbitrarily near to 1 by using $(1+\epsilon)$-spanners and taking the best of multiple runs.

Is there is some way to turn our Monte Carlo algorithm Las Vegas, or to allow $(1+\epsilon)$ approximation?

5 Discussion and Open Problems

5.1 Practicality

Are our algorithms of "practical" importance? This is not clear.

First, it depends on how much better than our worst-case bounds, our algorithms will typically perform. Second, it may be possible to combine our algorithmic ideas – which yield provable worst case bounds – with other, "heuristic" ideas, which may have drastic effects on the practicality of our algorithms, even though their effect on the worst case behavior is either nonexistent, or too complicated to analyze. Examples of possible heuristic ideas:

1. "Cleaning up" the "detours" induced by our spanner modification procedure by some kind of post-processing (perhaps a "local optimization").
2. Solving all sufficiently small subproblems arising in our algorithm via previous (heuristic local optimization, or exhaustive search) algorithms.
3. Choosing the quadtree split coordinates in some heuristic manner.

It seems impossible to tell how much effect these ploys will have, except by experiment.

A brief survey of the competition is below.

TST: In practice heuristic algorithms such as the Lin-Kernighan [29] [34] [14] local optimization procedure and Held-Karp lower bound [22] [37] will rapidly find a TST and a proof that it is within a small factor (usually a few percent) of optimality. No proof is known that the running time is always going to be rapid or that the approximation is always going to be good, but in practice it almost always is. Furthermore, programs that branch and bound (using high quality lower bounds arising from "polyhedral combinatorics" [26] [21] which empirically always seem to exhibit errors well below 1%) can find the optimal tour, and prove it, for TST problems with 1000s of sites [33][24]. These procedures undoubtedly consume asymptotically exponential runtime $(1+c)^N$, or quite likely even N^{cN}, for some $c>0$. But empirically c is remarkably small[25] ($c \approx 0.004$? Or even less?). The TST algorithms in this paper, and the previous ones by Arora, seem unlikely to be competitive with this level of performance in practice. For example – disregarding the constants in the Os – the $2^{(sd)^{O(d)}}$ factor in our runtime bound (EQ 8) when $d=2$ and $s=10$ (for 10% accuracy in the worst case), would be 2^{400}.

However, at the end of [1], Arora points out that a (slower form of) his algorithm may be interpreted as a local optimizer. (Ours too.) It is possible that this *interpretation* is of more practical value than our algorithms, because then the powerful competing techniques [33] [45] could be used as Arora's local optimizer! Also these techniques could be used to solve the small "TST with boundary conditions" subproblems arising in our algorithm.

If one does implement our TST algorithm, one may want to "clean up" the tour by performing, e.g., 2-opt or Lin-Kernighan as a postprocessing step. However, it is not known if this can be done in a polynomial number of steps, even in the plane[26].

SMT: The best computer program so far [45] can solve problems with 140 random[27] sites in 2D. See also [17] for a program which can solve 55-site 2D RSMT problems.

An impractical exact RSMT algorithm running in $N^{O(\sqrt{N})}$ time is described in [40]; recently the author (Smith) has generalized this to $N^{O(N^{1-1/d})}$ in d dimensions while at the same time making it much more practical.

Later note: A recent breakthrough (for practical purposes) is a program by D.Warme [44], in cooperation with P.Winter and M.Zachariasen, which can now solve 2000-site 2D SMT and 1000-site 2D RSMT problems. The worst case runtime behavior of their algorithm appears (!) to be doubly exponential: A^{B^N}.

At present, however, the best available program for SMT in ≥ 3D [40], can only solve 12-site SMTs.

Thus again – at least on the surface – it seems as though our algorithm cannot have much practical impact in 2D. Looking more deeply, again it may be possible by combining our techniques with the competing ones, to get a practical impact. By locally optimizing the SMT output by our algorithm with the aid of the Hwang $O(N)$-time 2D topology optimization algorithm described in [40] or the Smith multi-D topology optimization algorithm also described there, (see also [3]; also the exhaustive search algorithm of [45] may be used as a local optimizer by applying it to subtrees); and by using the competing exhaustive search algorithms to solve the small subproblems arising in our algorithm; it may be possible to find very good approximate SMTs even for very large N.

Since our current competition when $d \geq 3$ is abysmal, our approximate SMT algorithm may have direct practical value when $d=3$.

Finally, our approximate EC and M2M algorithms may be useful. We don't know of any competition here.

5.2 Open questions

Minimum Matching Can our $(1+\epsilon)$-approximation approach be made to work for MM still in $O(N \log N)$ time?

[24] The lowest cardinality of any as-yet inexactly solved TST problem in "TSPlib" [35] is 2103.

[25] Also, it should be possible to combine the $N^{O(\sqrt{N})}$ time exact 2D TST algorithm of [39] with the branch and bound programs to get the best of both worlds.

[26] Van Leeuwen and Schoone [27] show that performing any $\leq N^3/4$ elementary uncrossing operations will always uncross a TST (also MM, etc.) of N sites in general position in the plane. There is a version of our algorithm which may be thought of as producing a short crossing free tour but in which the paths between adjacent sites are not necessarily single line segments, but may instead contain numerous little detours. If these paths are "yanked taut" then our TST will become still shorter, but will no longer necessarily be crossing free.

[27] In an average time of <1 hour, but the hardest among 100 such problems required 34 hours. Problems in which the sites form an $n \times m$ grid give the program much more difficulty because there are then an exponentially large number of nearly equal SMTs. For such point sets, the program of [45] can only reach $nm = 35$.

Other models of computation: In models of computation permitting operations such as the "floor" function – models in which $o(N \log N)$ sorting is possible – is it possible to make our techniques run in $o(N \log N)$ time?

The authors believe that the answer to both of the above questions is "yes," at least if one is willing to accept "Monte Carlo" algorithms (which can fail with no indication of failure). In fact we have techniques to do it – except that we have not yet investigated these techniques in enough detail and with enough care to be sure of ourselves. If they work, we'll write them up in one or more subsequent papers.

Spanners: Improve the Arya spanner theorem further. First, one could try to replace the expression "$O(d)$" appearing in exponents in our bounds (§6) by something more precise, indeed in any particular fixed low dimension d one would like to know the exact answer. Second, one could investigate the interesting question, of just how much Steiner points can help. (We've shown in theorems 53 and 55 that they can help, tremendously.) Third, it might be possible to make better spanners still, where "better" is some notion tailored purely for the applications of the present paper.

Banyans: The dependence of our construction (§4) on d may be far from optimal. For example, our $(1+\epsilon)$-banyans use $(d/\epsilon)^{O(d)} d^{O(d^2)} N$ Steiner points, and perhaps only $(d/\epsilon)^{O(d)} N$ are really needed. Banyans seem to be of fundamental importance, so it is worth trying to understand them more precisely.

References

[1] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, 1997. Also available electronically on Arora's web page `http://www.cs.princeton.edu/~arora/publist.html` in an updated form.

[2] S. Arya, G.Das, D.M.Mount, J.S.Salowe, and M.Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 489–498, 1995.

[3] J.E. Beasley and F.Goffinet. A Delaunay triangulation-based heuristic for the Euclidean Steiner problem. *Networks*, pages 215–224, 1994.

[4] A. Borchers and D-Z. Du. The k-steiner ratio in graphs. *SIAM J. Computing*, 26(3):857–869, 1997.

[5] P.B. Callahan and S.R.Kosaraju. A decomposition of multi-dimensional point sets with applications to k-nearest neighbors and n-body potential fields. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 546–556, 1992.

[6] B.M. Chazelle. A faster deterministic algorithm for minimum spanning trees. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 22–34, 1997.

[7] N. Christophides. 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.* Academic Press, 1976.

[8] G. B. Dantzig. *Linear Programming And Extensions.* Princeton University Press, 1968.

[9] G. Das, S. Kapoor, and M. Smid. On the complexity of approximating Euclidean traveling salesman tours and minimum spanning trees. *Algorithmica*, 19:447–460, 1997.

[10] G. Das, G. Narasimhan, and J. Salowe. A new way to weigh malnourished Euclidean graphs. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 215–222, 1995.

[11] G. Das, P.Heffernan, and G.Narasimhan. Optimally sparse spanners in 3-dimensional Euclidean space. In *ACM Symposium on Computational Geometry*, pages 53–62, 1993.

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

[13] D.Z.Chen, G.Das, and M.Smid. Lower bounds for computing geometric spanners and approximate shortest paths. In *Canadian Conference on Computational Geometry*, pages 155–160, 1996.

[14] E.Aarts and J.K.Lenstra, editors. *Local search in combinatorial optimization.* J.Wiley, 1997.

[15] David Eppstein, Gary L. Miller, and Shang-Hua Teng. A deterministic linear time algorithm for geometric separators and its applications. *Fundamenta Informaticae*, 22:309–330, 1995.

[16] S. J. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, pages 153–174, 1987.

[17] U. Fössmeier and M. Kaufmann. Solving rectilinear Steiner tree problems exactly in theory and practice. 1997.

[18] Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM*, 38(4):815–853, 1991.

[19] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.*, 32(4):835–859, 1977.

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

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

[22] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[23] F.K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.*, 30(1):104–114, 1976.

[24] D. Karger, P. Klein, and R.E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J.ACM*, 42:321–328, 1995.

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

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

[27] J. Van Leeuwen and A.A. Schoone. Untangling a traveling salesman tour in the plane. In *Proc. 7th conf. on graphtheoretic concepts in computer sci. (WG 81), Linz, Austria*, pages 87–98, 1981.

[28] C. Levcopoulos and A. Lingas. There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees. *Algorithmica*, 8:251–256, 1992.

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

[30] L. Lovasz and M. Plummer. *Matching theory.* North-Holland, 1986.

[31] M.Ajtai, J.Komlos, and G.Tusnady. On optimal matchings. *Combinatorica*, 4:259–264, 1984.

[32] G.L. Miller, S-H. Teng, W. Thurston, and S.A. Vavasis. Separators for sphere packings and nearest neighbor graphs. *J. ACM (submitted)*, 1996.

[33] M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

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

[35] G. Reinelt. Tsplib – a travelling salesman program library. *ORSA J. Comput.*, 3:376–384, 1991. `http://junji.complex.hokudai.ac.jp/export/kebbe/index`.

[36] C.A. Rogers. Covering a sphere with spheres. *Mathematika*, 10:157–164, 1963.

[37] D.B. Shmoys and D.P. Williamson. Analyzing the Held-Karp TSP bound: A monotonicity property with application. *Info. Proc. Lett.*, pages 281–285, 1990.

[38] Michiel Smid. Lower bounds. *Lecture notes at Magdeburg*, 1997. `michiel@isgnw.cs.Uni-Magdeburg.DE`.

[39] W.D. Smith. *Studies in computational geometry motivated by mesh generation.* PhD thesis, Princeton Applied Math Dept., Princeton, NJ, 1988. University Microfilms International, order # 9002713.

[40] W.D. Smith. How to find Steiner minimal trees in d-space. *Algorithmica*, 7:137–177, 1992.

[41] Luca Trevisan. When Hamming meets Euclid: the approximability of geometric TSP and MST. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 21–29, 1997.

[42] P.M. Vaidya. Approximate minimum weight matching on points in k-dimensional space. *Algorithmica*, 4(4):569–583, 1989.

[43] P.M. Vaidya. An optimal algorithm for the all-nearest-neighbors problem. *Discrete and computational geometry*, 4:101–115, 1989.

[44] David M. Warme. A new exact algorithm for rectilinear steiner trees. In *International Symposium on Mathematical Programming, Lausanne, Switzerland*, 1997.

[45] P. Winter and M. Zachariasen. Euclidean Steiner minimal trees, an improved exact algorithm. *Networks*, 30(3):149–166, 1997.

6 Appendix: Improved analysis of Arya et al.'s spanners in dimensions $d \geq 2$

The latest and greatest results on Euclidean spanners in all dimensions – subsuming nearly every previous result ([5], [11]; [28]) are by Arya et al. [2]. But unfortunately, most of the "constant factors" in the [2] result and in the predecessor paper [10] had been left uncalculated. We will now analyze these factors, and the result is:

Theorem 45 (Improved Arya et al. Spanner Theorem) *For N sites in d-space and any fixed $\epsilon > 0$, there exists a $(1+\epsilon)$-spanner at most $(d/\epsilon)^{O(d)}$ times longer than the Steiner minimum tree, which has $(d/\epsilon)^{O(d)}N$ edges, maximum valence $(d/\epsilon)^{O(d)}$, and which may be constructed in $(d/\epsilon)^{O(d)}N + O(dN \log N)$ time. (The constants in our O's are independent of d, N, and ϵ.)*

Remark. Perhaps the "$d^{O(d)}$" factor in the length ratio bound could be improved to $2^{O(d)}$. But the "$\epsilon^{-O(d)}$" factor isn't improvable. To see this, use N random points on a unit d-sphere and take ϵ of order $N^{-1/(d-1)}$ (which is the mean distance from a point to its nearest neighbor), forcing any $(1+\epsilon)$-spanner (at least if it is not allowed to incorporate extra ["Steiner"] points) to have order N^2 edges of length ≥ 1. Meanwhile the MST of these points has length of order $N^{1-1/(d-1)}$, so the spanner is $\Omega(1/\epsilon)^d$ times longer than the SMT.

Remark. But, what are the correct constants to put in the $O(d)$s? Well, we think it is safe to say that these constants lie between 1 and 5. Furthermore, we have analysis (omitted because we are not yet 100% confident of it) of a new 2-dimensional spanner construction algorithm, which seems to indicate that in 2D, a $(1+\epsilon)$-spanner always exists, which is $O(\epsilon^{-2.51})$ times longer than the SMT. The example of N points uniformly spaced around a circle, with $\epsilon = 3/N$, shows that "2.51" cannot be improved beyond "2." However, the possibility still remains open that by extending the concept of "spanner graph" to now also allow extra "Steiner" vertices in the graph, we might be able to do better, perhaps even better than "2." This is explored in §6.1.

Remark. If one does not require near-linear runtime, and will be satisfied with runtime cubic in N, then a very simple "greedy algorithm" for constructing a $(1+\epsilon)$-spanner is known to work to yield a spanner at most $(d/\epsilon)^{O(d)}$ times longer than the Steiner minimum tree. Specifically, you consider candidate edges in increasing order of length, and add an edge (u, v) to the t-spanner if and only if the length of the shortest path from u to v in the (partially constructed old) spanner exceeds $(1+\epsilon)\cdot \mathrm{dist}(u, v)$.

In 2D, a simple construction does better than the Arya result in every way except for the fact that it will not necessarily be only a constant factor longer than the SMT:

Theorem 46 (2D Spanner Theorem) *For N sites in the plane, there exists a $(1+\epsilon)$-spanner which has $O(\epsilon^{-1}N)$ edges, maximum valence $O(\epsilon^{-1})$, and which may be constructed in $O(\epsilon^{-1}N\log N)$ time. (The constants in our O's are independent of d, N, and ϵ.)*

Proof sketch. This was shown in [39] so we only sketch the proof. The graph is the "directional nearest neighbor" graph DNNG in which every site P is joined to its nearest neighbor in a wedge with apex P and angular width $O(\epsilon)$. (We assume nearest neighbors are unique since the sites are in general position.) The distance approximation result is proven by consideration of a "greedy path." The runtime bound is shown by, for each of ϵ^{-1} direction ranges, finding the DNNG edges in that direction range in $O(N \log N)$ time with the aid of "directional voronoi diagrams" via a "plane sweep" paradigm. Everything has now been shown, *except* that the maximum valence in DNNG is not necessarily $O(\epsilon^{-1})$. The maximum out-valence obeys this bound, but the in-valencies need not. The solution to this, is to remove all the edges pointing into a vertex which has more than twice as much invalence as desired, except that edges which happen to be bidirected, are not removed. This clearly makes the invalencies obey the bound, and as is shown in section 4 of [2], will not cause ϵ to increase by more than a constant factor. □

Proof sketch for Improved Arya et al. Spanner Theorem 45: With all the $(d/\epsilon)^{O(d)}$'s omitted and replaced by "$O(1)$ when d is fixed," this was already proven in [2] and [10]. So all we plan to do is indicate how to tighten their analysis appropriately.

Definition 47 *Let $\kappa \geq 1$ be an integer. A set S of inter-site edges "satisfies the (κ, c)-isolation property" if: for each edge of length L there exists a hypercylinder of length and radius cL and axis a subinterval of that edge, where c, $0 < c < 1$, is a constant, such that each hypercylinder intersects at most κ edges.*

The main result of [10], as tightened by us, is the

Lemma 48 ((κ, c)-isolation Lemma) *If a set S of inter-site edges satisfies the (κ, c)-isolation property, then the total length of all the edges is*

$$O(\frac{\kappa d \log d}{c})\ \ell(SMT) \tag{29}$$

where SMT is the Steiner minimal tree of the sites and the constant in the "$O(d)$" is independent of N, d, and c.

Remark Actually, Das et al. [10] only proved this for $\kappa = 1$ (in which case we will speak of "c-isolation"), but the result for $\kappa > 1$ follows immediately from the $\kappa = 1$ result by considering the "interference digraph" of the edges. (An edge A "interferes with" edge B if edge A intersects B's hypercylinder.) The interference digraph has all outvalencies$\leq \kappa - 1$. Hence its undirected version has average valence $\leq 2\kappa - 2$ and is colorable with $\leq 2\kappa - 1$ colors by removing a vertex of valence $\leq 2\kappa - 2$, inductively coloring the remaining graph – which still has all outvalencies$\leq (\kappa - 1)$ – and then coloring the removed vertex. Now each color class of edges satisfies the plain c-isolation condition. Hence we may assume $\kappa = 1$ WLOG in what follows. Also WLOG we may then demand that all the hypercylinders be *disjoint* since that could then be accomplished by shrinking them by a factor of 2.

Two subsidiary results they needed are:

Lemma 49 (Long Steiner edge Lemma) *Any Steiner tree of N sites on a unit hypersphere in d-space, plus a site at the sphere center, must contain an edge longer than $\Omega((d \log d)^{-1})$. Here the constant in the "Ω" is independent of d and N.*

Proof sketch. There are two possible proofs of this. The first proof uses our lemma 36. We explore outward from the sphere center out to depth $k = \Omega(d \log d)$ edges deep into the SMT. If this exploration reaches the surface of the sphere, or even halfway there, we are done – an edge of length $\geq 1/(2k)$ must have occurred in the $\leq k$-edge path that got us there. But by the empty sphere lemma 36 with $r = 1/2$, we *must* have reached radius $\geq 1/2$, since the explored portion of the SMT has $2^k = d^{\Omega(d)}$ vertices since Steiner points have valency 3, and all the vertices must be Steiner since the unit sphere is by hypothesis empty – but lemma 36 says the radius-1/2 sphere contains only $d^{O(d)}$ Steiner points, which (with a sufficiently large constant in our definition of k) would be a contradiction.

The second proof is to follow the proof in Das et al. [10]'s section 2.2, but now making the numbers explicit. Let the site at the sphere center be regarded as the tree root. Suppose the portion of the SMT whose path distance to the root lies in the interval $(1-2\cdot 2^{-i}, 1-2^{-i})$ has no edge longer than $1/R_i$. The Das argument shows that

$$1+\log_2 R_i \geq \frac{R_{i-1}}{(d-1)2^i}. \qquad (30)$$

Also, if $2^{-i}R_i > \log_2 N$ for any i, a contradiction results (the SMT would have more than N leaves). The claim is that if R_1 is a sufficiently large constant times $(d-1)\log d$, then that will be sufficient to start the recurrence (30) growing explosively forever. Thus $R_1 = O(d \log d)$ is essential to prevent this superexponential growth (and the resulting contradiction), which proves our claim. □

Lemma 50 (Spherical cap covering lemma) *You can cover a sphere with spherical caps of angular radius θ using only $O(1/\theta)^{d-1}$ caps.*

Proof: e.g., see [36]. □

Proof sketch for c-isolation lemma 48: We follow the argument in Das et al. [10]'s section 3.1. Take the edges in S and divide them into nearly parallel (within, say, 20°) groups according to the direction cones of the Spherical cap covering lemma. We now consider only one such group of S edges, WLOG the one within 20° of being vertical. All the hypercylinders in this group are (WLOG if we pre-shrink our cylinders by a factor of 2 if necessary) disjoint. For each such group, we argue that its length is $O(d \log d)\cdot \text{length(SMT)}$. Because, there are 2 kinds of edges:

1. Edges such that the part of the SMT inside the hypercylinder joins both ends of the hypercylinder. In that case, clearly we can charge our edge length to the SMT length.

2. The other edges. In this case, consider the edge with the topmost top endpoint. Extend its hypercylinder infinitely far in both directions. Because our topmost endpoint must be connected in the SMT to the outside world, the SMT must include a point at the cylinder axis (namely, our edge's top endpoint) and on the cylinder's walls. In this case we can apply the Long Steiner edge Lemma 49 to see the relevant part of the SMT must contain a long edge, which we get rid of, replacing it with our edge instead (costing an order $d \log d$ factor) and then we continue on performing modifications to the SMT which keep it a connecting network but which now allow us to apply the argument again at the second topmost edge, the third, and so on.

Finally, note, in both cases above we also pay a *factor* $O(1/c)$ because the length and radius of the cylinder are cL for an edge of length L. Because the length and radius of the cylinder are equal, we in fact only needed $\theta = 20°$, (anyhow a constant) in the spherical cap covering Lemma 50, causing us to pay a factor (the cardinality of the cap covering) only $2^{O(d)}$. The product of all the factors we pay is (29). □

Now (to proceed finally with the proof of the Improved Arya Spanner Theorem) [2] based their results on a construction of "s-separated pairs" by [5], who showed how to compute a complete set of $O(s^d N)$ s-separated pairs in $O(N \log N + s^d N)$ time and $O(s^d N)$ space.

Definition 51 *Two sets A and B of points are "s-separated" if*

$$s \cdot \min\{\text{diam}A, \text{diam}B\} \leq \text{dist}(A, B). \qquad (31)$$

Definition 52 *A "complete set of M s-separated pairs" for an N-point set, is a set of M subsets A_1, A_2, ..., A_M, and another set of M subsets B_1, B_2, ..., B_M, of the points, such that for every pair of points a, b, there exists a unique j so that $a \in A_j$, $b \in B_j$, and (A_j, B_j) is s-separated.*

[2] observed that a complete set of s-separated pairs could be converted into a $(1+\epsilon)$-spanner, not necessarily of small total length, where $s = O(d/\epsilon)$.

[2] (in their section 5) then showed how to modify these pairs by moving some of the "box representatives" to the sides of their boxes, and then by deleting certain long edges in cones of angular radius $O(\epsilon)$, to get a shorter $(1+\epsilon)$-spanner. All of these conversions could be accomplished in time depending only linearly on the number of edges in the spanner. By using packing arguments one may see that if the edges in this spanner are partitioned into direction-cone classes according to the caps in a spherical cap covering as in the Spherical cap covering lemma 50 (but now using angular radius $\theta = O(\epsilon)$ rather than 20°) each class necessarily will be (κ, c)-isolated, with $c/\kappa = (\epsilon/d)^{O(d)}$. Now applying the (κ, c)-isolation lemma 48 completes the proof. □

6.1 Improving spanners by allowing "Steiner points"

One could extend the definition of "spanner" to allow extra "Steiner" vertices to exist. The fact that this could permit tremendously shorter spanners, follows from the examples in theorems 53 and 54, and the algorithmic improvement in theorem 55.

Theorem 53 (2D counterexample Theorem) *If $N = 2^p$, $p \geq 1$, here is an example of a 2D $2N$-point set in which a $(1+\epsilon)$-spanner with Steiner points exists that is order $\epsilon^{-1}/|\log \epsilon|$ times shorter than any $(1+\epsilon)$-spanner without: Space N points uniformly along the East and West sides of a unit height rectangle of width N, and choose $\epsilon = 0.75/((N-1)N)$.*

Proof. Any Steinerless spanner must connect every point on the East side directly to every point on the West side for total length of order N^3, i.e. order ϵ^{-1} times the length (which is $N+2$) of the MST.

However, draw a complete binary tree (we've assumed N is a power of 2) with root at the rectangle center and leaves at the N East side points, in such a way that all edges of this tree are line segments with slope $\pm 1/N$. (Also mirror this tree about the vertical midline.) The resulting network will be a spanner with approximation ratio $\leq \sqrt{1+N^{-2}} < 1+\epsilon$. Its total length will be $O(N \log N)$, which is $|\log \epsilon| \cdot \ell(MST)$. □

This construction can be generalized to dimensions $d \geq 3$. Distribute N points randomly in the two $(d-1)$-balls that form the end caps of a d-dimensional hypercylinder of unit radius and length L of order $N^{1/(d-1)}$.

Theorem 54 (General-d counterexample Theorem) *If $d \geq 3$ is fixed, then for the d-dimensional point set described above, in the limit where $N \to \infty$: When ϵ is of order $N^{-2/(d-1)}$, any $(1+\epsilon)$-spanner without Steiner points must have length at least $\epsilon^{-\Omega(d)}$ times longer than the shortest $(1+\epsilon)$-spanner which has Steiner points.*

Proof omitted.

Theorem 55 *The length bound in the $(1+\epsilon)$-spanner construction of [2] may be improved by a factor of $\epsilon^{\Omega(1)}$ while leaving the approximation behavior unaffected by putting Steiner points into the construction.*

Proof sketch for the case $d=2$ only: The key observation in the most naive version of the Arya et al. [2] construction was that by joining set representatives by an edge, by use of a complete set of s-separated pairs one could get a $1+O(d/s)$-spanner. We can instead join well separated pairs by a tree-like object with order s leaves as in the Counterexample Theorems above, to get a spanner with approximation error of order s^{-2} instead of s^{-1} when $d=2$. Now in [2]'s less-naive constructions whose purpose was to get a *short* spanner, they perform various alterations and make various arguments based on cones of angles and c-isolation. These arguments may also be made in our case, although we have to take narrower angular cones (of angular radius $O(s^{-2})$ now, not s^{-1}) due to our desire to get squared approximation error. At the end of the argument, we have still saved an $s^{\Omega(d)}$ factor in some parts of the argument (versus just squaring ϵ and then using the plain Arya construction), although not in other parts, resulting (since the factors multiply) in a genuine savings. □

Although the above results suggest that "steinerizing" spanners could improve them drastically, the below result shows that the improvement, at least for some point sets, can never be drastic enough to impugn the optimality of the "$\epsilon^{-O(d)}$" factor in our improved Arya Spanner Theorem 45.

Theorem 56 (Steinerized spanners cannot be too short) *For random points in d-space (arising from a d-dimensional Poisson process), any $(1+\epsilon)$-spanner, even a steinerized one, must be (with high probability) at least of length $\Omega(1/\epsilon)^{(d-1/d)/2}$ times the length of the SMT.*

Proof sketch. Consider points distributed in d-space according to a unit density Poisson process.

Definition 57 *The "R-ellipsoid" with foci A, B is the locus of points P such that $\mathrm{dist}(A,P)+\mathrm{dist}(B,P) \leq R\,\mathrm{dist}(A,B)$.*

Let $R = 1+\epsilon$. If we can find a large number of R-ellipsoids, every two of which are almost entirely *disjoint*, then any (even steinerized) R-spanner must be at least as long as the sum of the lengths of these ellipsoids, because any R-spanner path from A to B has to be entirely enclosed inside the R-ellipsoid of A and B.

Typical points P have order $\epsilon^{-(d-1)/2}$ points at a distance from them of order $\epsilon^{-(d-1)/(2d)}\sqrt{d}$. The $(1+\epsilon)$-ellipsoids with foci at these points and at P are each empty with a probability of order 1, since they have d-volume of order 1. In fact by changing the constants we can make the emptiness probability become $1-O(c^d)$, because the ellipsoid d-volumes are of order c^d, for any constant c, $0<c<1$. Throw out ellipsoids that are not empty. We claim a large fraction of these ellipsoids with common focus P are disjoint except for comparatively small regions (of diameter much smaller than their length) located near P. (This may be seen by considering them angularly, viewed from their common focus.) Throw out ones that are not (more precisely, find a large independent set among them, which can be done by a simple min-valence greedy algorithm because the average valency of the intersection graph of the ellipsoids is very small).

The total volume fraction of d-space occupied by the ellipsoids we've just described is $O(c^d)$ (a similar result is also true in any $(d-1)$-dimensional subspace). This means that the fact that our points are random, which means any two ellipsoids are completely independent (unless they share a foci, which is a case we dealt with specially) means that the probability that a given ellipsoid is intersected by any other independent one, is very small. This may be used to see that, if c is sufficiently small, then with high probability a large fraction of the ellipsoids we described in fact are disjoint (except for small regions near common foci, which don't matter).

This leads to the conclusion of the theorem. (Note, the MST of points from a unit density Poisson process has length $O(\sqrt{d})$ per point, by, e.g. considering joining every point to its nearest neighbor that lies below it.) □