Approximating geometrical graphs via "spanners" and "banyans"

Satish B. Rao & Warren D. Smith
NECI, 4 Independence way, Princeton NJ 08544
{satish,wds}@research.NJ.NEC.COM

Abstract

The main result of this paper is an improvement of Arora's method to find $(1+\epsilon)$ approximations for geometric NP-hard problems including the Euclidean Traveling Salesman Problem and the Euclidean Steiner Minimum Tree problems. For fixed dimension d and ϵ, our algorithms run in $O(N \log N)$ time.

An interesting byproduct of our work is the definition and construction of *banyans*, a generalization of graph spanners. A $(1+\epsilon)$-banyan for a set of points A is a set of points A' and line segments S with endpoints in $A \cup A'$ such that a $1+\epsilon$ optimal Steiner Minimum Tree for any subset of A is contained in S. We give a construction for banyans such that the total length of the line segments in S is within a constant factor of the length of the minimum spanning tree of A, and $|A'| = O(|A|)$, when ϵ and d are fixed.

In this abbreviated paper, we only provide proofs of these results in two dimensions. The full paper on WDS's web page (`http://www.neci.nj.nec.com/homepages/wds`, click "NECI technical reports") extends the techniques to higher dimensions, proves some new facts and clarifies some old facts about spanners, and also gives approximation algorithms for minimum matching, edge cover, rectilinear Steiner minimum tree, and minimum 2-matching.

1 Introduction

We improve a method of Arora [1] for approximating various geometric NP-hard problems, such as the traveling salesman problem and Steiner Minimum Tree problem.

For example, Arora's algorithm, given N points in a Euclidean d-space, $d \geq 2$, produces, with probability $1/2$, a $(1+1/s)$-approximate traveling salesman tour (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 (and $O(N)$ space) algorithms for this problem for any fixed s and d. In more detail, we state the following main result.

Main Result I. In d dimensions, our methods yield $1+1/s$ approximation algorithms for TST or SMT that succeed with probability $1/2$ in a Monte Carlo sense[2] in time

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

and

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

space, or in a Las Vegas sense[3] in time

$$2^{(sd)^{O(d)}} N + O(dN \log N), \tag{3}$$

or a deterministic algorithm with runtime

$$2^{(sd)^{O(d)}} N + (sd)^{O(d)} N \log N, \tag{4}$$

in both of the latter cases consuming

$$(sd)^{O(d)} N + 2^{(sd)^{O(d)}} \log N \tag{5}$$

space.

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

1.1 New Ingredients

The main new idea for a TST algorithm is to 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 [7] showed that spanners could be found with sum of the edge lengths at most $a_d(\epsilon)\ell(MST)$ where $\ell(MST)$ is the length of the minimum spanning tree of the sites. They did not specify the function $a_d(\epsilon)$. We redo the analysis of Arya et. al. [2] to show in the full paper that $a_d(\epsilon) = (d/\epsilon)^{O(d)}$.

The full paper also studies extending the notion of Euclidean spanner graphs to allow Steiner points. Specifically, we prove 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 the full paper 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.

More significantly, we introduce the following generalization of spanners to extend the approximation algorithm for TST's to Steiner Minimum Trees.

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.*

Main Result II. For fixed ϵ and d, a $(1+\epsilon)$-banyan of any set of N sites in d-space exists, which

[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.

[2] The algorithm does not determine whether it is successful or not.

[3] The algorithm either gives a correct answer or informs the user that it failed.

- 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.

In §3, we prove this result for $d = 2$. For $d \geq 3$ see the full paper.

EC	(Minimum) edge cover
MM	Minimum matching
M2M	Minimum 2-matching
MST	Minimum spanning tree
RSMT	Rectilinear Steiner minimum tree
SMT	Steiner minimum tree
TST	(Minimum) Traveling salesman tour

Table 1: Geometrical graph acronyms we use in this paper.

1.2 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* to reconnect all of the cut edges (figure 2).

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

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

The procedure above and its analysis arose in Arora's proof of his main structure theorem [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 because Arora considered applying (but didn't apply) 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 box boundary in a randomly shifted 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.

1.3 Applicability

Roughly speaking, our methods apply to a geometrical graph G if it's sparse, simply defined[4], 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 MM and k-vertex subgraph variants of the MST 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, for any particular graph it *might* be possible to overcome its shortness and non-roundability. In the full paper, we show how to approximate EC and M2M to within $1+1/s$ with our usual time and space bounds, despite the fact that they both can be much shorter than the MST. This gives some hope that it might be possible to handle MM in the future, although we do not know how at present. Also in the full paper, we show that our techniques nevertheless can approximate MM to within a factor of $2^{O(\sqrt{d})}$ in $O(d^{O(d)} N \log N)$ *Monte Carlo* time.

[4] 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.

1.4 How close are we to optimal speed?

The $N \log N$ behavior of our TST, SMT, spanner, and banyan runtime (4) when s and d are held fixed is optimal [6][5] in an "algebraic computation tree" model[6].

It is very likely that (4)'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 [25] showed that approximating TST to within any sufficiently small constant factor for N sites in $\log N$ dimensional space, is NP-hard. 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.

1.5 Related Work on Geometrical Graph algorithms

Christophides [5] 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 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) [22]. This approximation ratio 1.5 is still best known for TST in arbitrary metric spaces for approximation algorithms running in polynomial time.

Smith [23] 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 [24] he extended this to handle RSMT.

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. RSMT in the Euclidean plane is at most 3/2 times shorter than the L_1 MST [15]; SMT in the Euclidean plane is at most $2/\sqrt{3}$ times shorter than the MST [8]; and these factors are both tight.

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

1.6 Organization of the paper

In §2 we describe how to approximate the traveling salesman tour in the plane; §3 describes banyans and their use for approximating SMTs; and §4 concludes.

2 Approximating the TSP in the plane

We use lemmas stated in [1] and [16] about $(1+\epsilon)$-approximate traveling salesman tours for N sites in the plane. They all generalize to d-space with $d > 2$, but in this abbreviated paper we'll sometimes only state them with $d = 2$.

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.

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 integer *shifts* 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.

Then we start with a square that includes the whole resulting grid, and then divide it into four. We then recurse on the boxes that contain more than 1 site. A shifted quadtree dissection of a set of points on the grid with shift (a,b) is a quadtree dissection where each square is shifted to the left by a and down by b for integer $a, b \in [-L/2, L/2]$. (Notice that the original center grid is entirely covered by the set of boxes in the shifted boxes, although parts of the expanded grid may not be.)

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*

$$\sum_{l:\text{horizontal}} t(G,l) + \sum_{l:\text{vertical}} t(G,l) \le \sqrt{d}\ell(G), \qquad (6)$$

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

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.*

---

[5] Actually [6] did not prove this for spanners and banyans. However by applying any $o(N \log N)$ MST algorithm [4] to either of these graphs we would recover an approximate MST, hence by [6]'s $\Omega(N \log N)$ lower bound for approximate MSTs, the result follows.

[6] 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 in an honestly purely algebraic manner still with $O(N \log N)$ runtime.

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 $\leq 2q$ infinitesimally shifted versions of l, $\leq q$ 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 $r \geq \kappa$, that is in expectation at most $(1+\frac{g\sqrt{8}}{r+1-\kappa})^2$ times longer than G.*
- *An r-light graph G', for any r with $r \geq \kappa$, 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, 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 approximately 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 quadtree shift (a,b) is

$$\leq \frac{2^i}{L}. \tag{7}$$

We patch the graph by using the procedure of figure 1 on each line. Recall that the actual shifted dissection consists of lines of length $2L$.

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

ARORA-MODIFY(l,i,b)
(l is a vertical box 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.

Figure 1: Patching a line.

patch replaces $\geq r+1$ crossings by κ crossings, we have for each vertical shift b

$$\sum_{j\geq 0} c_{l,j}(b) \leq \frac{t(G,l)}{r+1-\kappa}. \tag{8}$$

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

$$g\sum_{j\geq i} c_{l,j}(b)\cdot\frac{L}{2^{j-1}}. \tag{9}$$

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

$$\begin{aligned} g\sum_{i\geq 0}\frac{2^i}{L}\sum_{j\geq i} c_{l,j}(b)\frac{L}{2^{j-1}} &= g\sum_{j\geq 0}\frac{c_{l,j}(b)}{2^{j-1}}\sum_{i=0}^{j} 2^i \\ &< 4g\sum_{j\geq 0} c_{l,j}(b) \\ &\leq \frac{4g\ t(G,l)}{r+1-\kappa} \end{aligned} \tag{10}$$

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) is at most the original length plus

$$\sum_{l}\frac{4g\,t(G,l)}{r+1-\kappa} \leq \frac{g\sqrt{8}}{r+1-\kappa}\ell(G). \tag{11}$$

The horizontal lines of the quadtree can be dealt with similarly. But the two sets of lines can *interact* because patches along vertical quadtree box 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 $(1+\frac{g\sqrt{8}}{r+1-\kappa})^2$ 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 $a_1, a_2, \ldots a_k$ be real with $0 < a_i$ and $\sum_{i=1}^k a_i = S$. Then*

$$\prod_{i=1}^{k}(1 + a_i) \leq (1 + S/k)^k < \exp S. \tag{12}$$

Proof sketch: (12) 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 amplication factors" caused by the patchings[7]) 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).*

[7]We may assume WLOG (since the patching lemma remains valid) that there is a graph crossing infinitesimally close to one endpoint of every quadtree box boundary line segment we desire. Then, the extra crossings caused on child edges by parent patchings will not cost us anything.

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[8] 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 $\kappa = 1$. See figure 2.

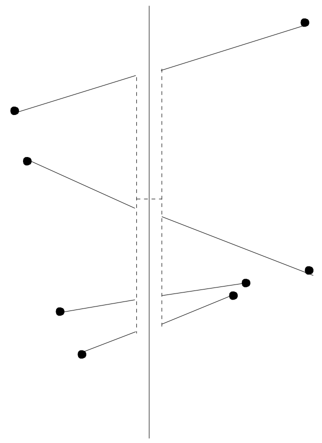

Figure 2: Proof of the patching lemma for spanners.

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

[8]connected and with every valency even

least cs^4 for an appropriate constant c, we ensure 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[9] of size $O(Ns)$.

2. Find a t-spanner graph S of the N sites according to [2]. 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 $O(s^3N)$ time since the total number of crossings is $O(s^3N)$ 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 $O(s^3N\log N)$ 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[10] 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.)

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 the full paper, 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 $\ell(S) \leq a_d(\epsilon)\ell(MST)$, and of course $\ell(MST) \leq \ell(TST) \leq 2\ell(MST)$. (In the present analysis $d=2$, of course.)

[9] 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.

[10] 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 [23].

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 box, with $\leq 2^{O(r)}$ 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$\geq 1/2$ 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 the full paper).

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 $\geq 1/2$) 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 $O(ds^3 N/\delta)$ 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 $O(dN \log N)$ time. So assume 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 $j \in \{0, 1, 2, \ldots, 1 + \log_2 L\}$ 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, 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 box boundary at most $O(s^{O(1)})$ times,*
- *π only crosses any box 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 boxes 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 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 for SMT. The full paper also does RSMT, EC, and M2M. We still don't know how to get a $(1+\epsilon)$ approximation algorithm for MM. But the full paper gives 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.

3.1 Banyans

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

1. Find a $(1+O(\epsilon))$-spanner graph as in [2].
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.

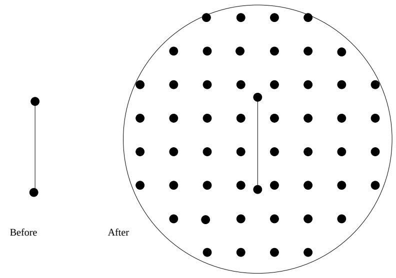

Figure 3: Conversion of a spanner edge into a set of Banyan points. The circle has radius $\kappa_d L/\epsilon$ where L is the length of the edge. The grid has spacing $\epsilon L/\kappa_d$.

5. Finally, output a $(1+O(\epsilon))$ spanner of both the original sites and the Steiner candidate points.

Theorem 24 (Banyan theorem) *Let $0 < \epsilon \leq 1$. If κ_d is a sufficiently large (it will suffice if $\kappa_d = d^{O(d)}$ and $\kappa_2 = 303$) then the graph described above is a $(1+\epsilon)$-banyan. If d and ϵ are fixed: The algorithm described above runs*[11] *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 produced will also have bounded valencies and will be at most a constant factor longer than the SMT of the sites.*

Theorem 24 will be proven in 2D in the next subsection.

Remark. 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.)

3.2 Steiner minimal trees (SMTs) in 2D

Assuming one accepts theorem 24, the rest is easy.

Corollary 25 *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 $(1+\epsilon)$.*

This is since 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 26 *"Steiner points" are vertices of the SMT that are not members of the original set of N "sites."*

[11] 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.

Proof of theorem 24 in 2D. We first remark that we don't need to prove theorem 24 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 $\frac{1}{2}\ell(MST) < \ell(SMT) \leq \ell(MST)$. The length of the SMT of any site subset does not exceed the length of the SMT of the full set[12].

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, i.e. don't cross.

Fact 5: (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.

The above facts 0-5 were shown in [12]. All of them are valid in general dimension except for fact 5.

Lemma 27 (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 $\leq L$. 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 [11], but we include it here for completeness. Refer to figure 4 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 5) connected by an SMT path to its apex; and V_A must lie inside hexagon H_A due to fact 4. □

Lemma 28 (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 24 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 24 is proven since AB must be in the spanner (or a path closely approximating it). So assume A is Steiner.

[12]MST does not have this property.

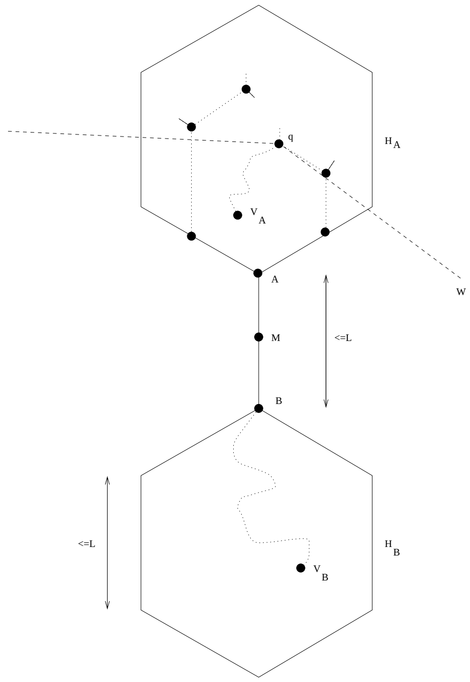

Figure 4: Proof of the hexagon property. 120° wedge W must include a site connected to q by a path contained entirely inside it. All SMT edges shown are $\leq L$ long; the regular hexagon sides are L.

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

$$D = (1 + \epsilon/2)S = 101L + 200L/\epsilon + \epsilon L/2. \qquad (13)$$

centered at M. (See figure 5.) Ignore everything outside Q.

By lemma 28 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 this case theorem 24 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

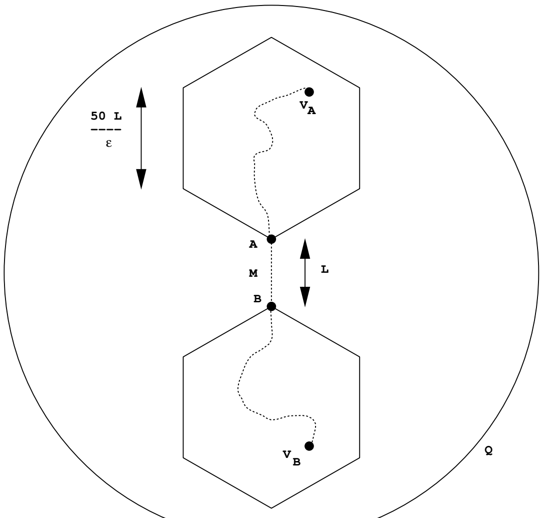

Figure 5: Proof of banyan construction's approximation property in 2D.

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 24 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 Discussion and Open Problems

4.1 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 analyse. 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 [19] [21] [9] local optimization procedure and Held-Karp lower bound [14] [22] 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" [17] [13] 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 [20][13]. 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[14] ($c \approx 0.004$? 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 4) 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 [20] [27] 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[15].

SMT: The best computer program so far [27] can solve problems with 140 random[16] sites in 2D. See also [10] 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 [24]; 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 [26], 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 $\ge$ 3D [24], 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

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

[14] Also, it should be possible to combine the $N^{O(\sqrt{N})}$ time exact 2D TST algorithm of [23] with the branch and bound programs to get the best of both worlds.

[15] Van Leeuwen and Schoone [18] show that performing any $\le N^3/4$ elementary uncrossing operations will always uncross a TST 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.

[16] 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 [27] can only reach $nm = 35$.

algorithm described in [24] or the Smith multi-D topology optimization algorithm also described there, (see also [3]; also the exhaustive search algorithm of [27] 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 $d = 3$.

4.2 Open questions

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

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 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. (The full paper showed 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 (§3) 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] 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.

[5] 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.

[6] 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.

[7] 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.

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

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

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

[11] 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.

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

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

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

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

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

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

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

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

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