Approximation Schemes for Degree-restricted MST and Red-Blue Separation Problem

Sanjeev Arora*
Department of Computer Science
Princeton University

Kevin Chang
Department of Computer Science
Yale University
51 Prospect Street
New Haven, CT 06511
(203)-436-1252 (telephone)
(203)-432-0593 (fax)
kevin.chang@yale.edu

Abstract

We develop a quasi-polynomial time approximation scheme for the Euclidean version of the Degree-restricted MST problem by adapting techniques used previously by Arora for approximating TSP. Given n points in the plane, $d = 3$ or 4, and $\epsilon > 0$, the scheme finds an approximation with cost within $1 + \epsilon$ of the lowest cost spanning tree with the property that all nodes have degree at most d.

We also develop a polynomial time approximation scheme for the Euclidean version of the Red-Blue Separation Problem, again extending Arora's techniques. Given $\epsilon > 0$, the scheme finds an approximation with cost within $1 + \epsilon$ of the cost of the optimum separating polygon of the input nodes, in nearly linear time.

*Supported by David and Lucille Packard Fellowship and NSF Grants CCR-0098180 and 0205594. Work done partially while visiting the CS Dept at UC Berkeley

1 Introduction

In the *degree-restricted minimum spanning tree problem* we are given n points in $\mathbb{R}^2$ (more generally, $\mathbb{R}^k$) and a degree bound $d \geq 2$, and have to find the spanning tree of lowest cost in which every node has degree at most d. The case $d = 2$ is equivalent to traveling salesman and hence NP-hard. Papadimitriou and Vazirani [16] showed NP-hardness for $d = 3$ in the plane, and conjectured that the problem remains NP-hard for $d = 4$. The problem can be solved in polynomial time for $d = 5$ since there always exists a minimum spanning tree with degree at most 5. We are interested in approximation algorithms for the difficult cases, both in $\mathbb{R}^2$ and $\mathbb{R}^k$. (In $\mathbb{R}^k$ the degree bound on the optimum spanning tree is of the form $\exp(\Theta(k))$, so all degrees less than that are interesting cases for the problem.) This problem is the most basic of a family of well-studied problems about finding degree-constrained structures; see Raghavachari's survey [17].

An *approximation scheme* for an NP-minimization problem is an algorithm that can, for every $\epsilon > 0$, compute a solution whose cost is at most $(1+\epsilon)$ times the optimum. If the running time is polynomial for every fixed ϵ, then we say that the approximation scheme is a *Polynomial Time Approximation Scheme* (PTAS), and if the running time is not quite polynomial but $n^{\text{poly}(\log n)}$ then we say the approximation scheme is a *Quasipolynomial Time Approximation Scheme* (QPTAS).

We now know that many geometric problems have PTASs. Arora [1], and independently, Mitchell [15] showed the existence of PTASs for many geometric problems in $\mathbb{R}^2$, including Traveling Salesman, Steiner Tree, k-TSP, k-MST. (Arora's algorithms also extend to instances in $\mathbb{R}^k$ for any fixed k.) Later, the running time of many of these algorithms was improved to nearly linear (Arora [2], Rao and Smith [18].) Similar PTASs were later designed for other geometric problems, such as k-median and facility location (Arora, Raghavan, Rao [5], Kolliopoulos and Rao [12]), minimum latency (Arora and Karakostas [4]), minimum k-connected subgraph (Czumaj and Lingas [9]), etc. For a current survey of approximation schemes for geometric problems, see Arora [3].

The above-mentioned survey notes that all these results use similar methods. Underlying the design of the PTAS is a "Structure Theorem" about the problem, demonstrating the existence of a near-optimum solution with very local structure— namely, it is possible to give a recursive geometric dissection of the plane such that the solution crosses each square in the dissection very few times. (A simple dynamic programming can optimize over such solutions.) The Structure Theorem is proved by showing that an optimum solution can be modified —by breaking it up locally and "patching it up" so as to not greatly increase cost— to have such local structure.

The survey goes on to note that this method of proving the Structure Theorem breaks down when the optimum solution has to satisfy some topological conditions. Two examples were given: the degree-restricted MST, and minimum-weight Steiner triangulation. Attempts to break the optimum solu-

tion and patch up the effects seems to have a rippling effect as we try to reimpose the topological constraint (the degree constraint, or the fact that the solution is a triangulation); the ensuing nonlocal changes are difficult to reason about. [1]

In this paper we present a QPTAS for the degree-restricted MST problem in the plane and in $\mathbb{R}^k$, for any constant k. The running time in $\mathbb{R}^2$ is $n^{O(\log^5 n)}$. For any $d > 2$, our algorithm, generalized to $\mathbb{R}^k$, runs in time $n^{O(\log^{2k+1} n)}$. The first approximation algorithms for this problem were due to Khuller, Raghavachari and Young [11], who gave a 1.5-approximation for $d = 3$ and a 1.25-approximation for $d = 4$ in $\mathbb{R}^2$. For $d = 3$, they gave a 5/3-approximation in $\mathbb{R}^k$, but have no such result for $d > 3$. Chan [8] later improved these approximations to 1.40 for $d = 3$ and 1.14 for $d = 4$. For $d = 3$, he gave a 1.63 approximation in $\mathbb{R}^k$.

We also present a PTAS for another problem with a topological flavor, the *Red-Blue Separation* problem. In this problem we are given a set of points in the plane, where each point is labelled either red or blue. We desire the simple polygon with lowest aggregate edge length that contains all the red points and no blue points. This problem is NP-hard, and its PTAS is easier to design. The previous best result for the Red-Blue Separation problem was an $O(\log n)$ approximation algorithm due to Mata and Mitchell [14]. Mitchell's aforementioned techniques for approximating TSP do in fact apply to the Red-Blue separation problem and yield a PTAS (private communication from Joe Mitchell), but such an algorithm would be less efficient than ours.

2 Degree-Restricted MST

We start by defining the recursive scheme for dissecting the instance, namely, a $1/3 : 2/3$-*tiling*. It differs from the dissection used in other geometric PTASs in that it is not picked randomly at the start. Instead, the algorithm searches for a suitable tiling using dynamic programming. This search goes hand-in-hand with the dynamic programming used to find the near-optimum solution.

Let *bounding box* be the smallest square around our instance, and let L denote the length of each side. We assume by rescaling distances that L is an integer and is n^3. The rationale behind choosing n^3 is two-fold: the n^3 bound will limit the depth of the recursive dissection (the 1/3:2/3-tiling to be defined later) and will prove useful for the base case of the dynamic programming algorithm (to be discussed later.) Clearly, the optimum tree has cost $\mathrm{OPT} \geq L$.

In any rectangle of length l and width w a *line separator* is a straight line segment, parallel to the shorter edge of the rectangle, that lies in the middle third (i.e., its distance from each of the shorter edges is at least $l/3$.)

The $1/3 : 2/3$-*tiling* of our instance is a binary tree of rectangles, whose root is the bounding box. Each tree node is a rectangle, and its two children

[1] We note that the salesman problem also involves a topological constraint on the solution, namely, all degrees are 2. However, then the solution is 2-connected and a simple Patching Lemma —see Lemma 13— holds. This fact does not generalize to degree-restricted spanning trees, where we allow degrees of 1.

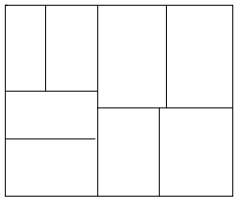

Figure 1: A depth three $1/3 : 2/3$-tiling.

are obtained by dividing the rectangle using some line separator. See Figure 1. We stop the partitioning when the rectangle's larger side has length at most L/n^2. (Recall, $L = n^3$ is the size of the root square.) Clearly, the depth of the tree is $O(\log n)$. Note that the number of distinct $1/3 : 2/3$-tilings could be exponential because the partitioning is free to pick any line separator at each step.

We will also associate with the tiling a set of *portals*. We designate an integer $m > 0$ as the *portal parameter*. Along each line separator used in the tiling, we place m evenly spaced points known as *portals*. A set of edges (for example, a spanning tree or a salesman tour) is called *portal respecting* if the edges cross each line separator only at its portals. The set of edges is called *r-friendly* with respect to a 1/3:2/3-tiling if each line separator is crossed by at most r edges. The set of edges is *(m, r)-light* with respect to a 1/3:2/3-tiling if it is portal respecting and r-friendly.

Throughout, d denotes the degree bound; in the plane the interesting cases are $d = 3, 4$.

THEOREM 1 (STRUCTURE THEOREM FOR DRMST) *There exists a $1/3 : 2/3$-tiling and a degree-restricted spanning tree with cost at most $(1 + \epsilon)OPT$, such that the tree is (m, r)-light with respect to this 1/3:2/3-tiling. Here $m = O(\log n/\epsilon)$ and $r = O(\log^5 n)$.*

2.1 Dynamic Programming Algorithm

The Structure Theorem leads to a dynamic programming algorithm that is a QPTAS for the problem. The dynamic programming is fairly standard and we describe it now. The objective of the algorithm is to build an r-friendly degree-restricted spanning tree and its corresponding tiling, such that the cost of the tree is within a factor of $(1 + \epsilon/3)$ of the lowest cost r-friendly degree-restricted spanning tree.

The basic subproblems solved by the dynamic programming involve the following inputs: (a) A rectangle, R. (b) A set of k edges ($k \leq 4r$) out of all possible $\binom{n}{2}$ edges $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ that cross the boundary of the rectangle. In other words, u_i's lie inside the rectangle and v_i's are outside. Any u_i, v_i could occur in multiple edges, subject of course to the degree constraint. (c) A tree on $\{u_1, \ldots, u_k\} \cup \{v_1, v_2, \ldots, v_k\}$ that includes the edges in (b). This tree forms a "template" for how the final solution must connect up the edges.

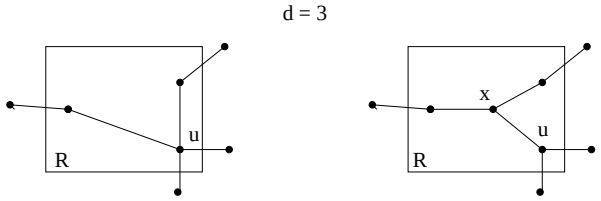

Figure 2: On the left, the template for the subproblem in rectangle R. On the right, a solution to the instance inside R. Note that the solution uses an internal node x to satisfy the degree bound on all vertices. If no such x existed, then the template would not be realizable, since the degree bound would not be satisfiable at node u.

Note that given (a), (b) and (c), the solution inside the rectangle is independent of the solution outside the rectangle. The basic subproblem of the dynamic programming algorithm is then to find an optimum r-friendly degree-restricted forest in R (i.e. a forest such that all nodes have degree at most d), such that if (u_i, u_j) is an edge in the template, then there is a path connecting u_i and u_j in the forest, and the forest contains every node inside R. It is helpful to note that the template itself need not explicitly capture the degree restrictions of the final solution (i.e. it is not a degree-restricted spanning tree), but rather just the connections between nodes. If such a forest cannot be realized without violating the degree restriction, then this template is rejected (and of course no r-friendly degree-restricted spanning tree may have such a local structure.) See Figure 2.

The solution to the instance defined by (a), (b), (c) is the portion of the desired degree-restricted spanning tree that lies inside the rectangle. First, we show that there are $\binom{n}{4}$ choices for (a). We can assume –by maximally "shrinking" the rectangles in the 1/3-2/3-tiling– that every line separator passes through a node. Hence, the number of combinatorially distinct rectangles in (a) is at most $\binom{n}{4}$, the number of 4-tuples of nodes. Consider next (b). Observe that the number of possible choices for k edges is $\binom{n^2}{k}$, and k can range from 0 to $4r$; thus the number of choices for (b) is $\sum_{k=0}^{4r} \binom{n^2}{k}$. Lastly, consider the number of choices for (c), which is at most the number of trees on $8r$ vertices. By a theorem of Cayley, we know that the exact number of such trees is $(8r)^{8r-2}$. Thus, the total number of subproblems (and hence the size of the dynamic programming table) is at most $\binom{n}{4} \times (\sum_{k=0}^{4r} \binom{n^2}{k}) \times (8r)^{8r-2}$, which is at most $n^{O(\log^5 n)}$.

To solve a subproblem we proceed in the usual bottom-up manner. The base case involves rectangles whose each side is at most L/n^2. We solve these arbitrarily, subject of course to the constraints imposed by the template (we argue below that using an arbitrary solution does not greatly affect the final cost.) For any other rectangle, we try all possibilities for the line separator; for each such partition into two smaller rectangles we try all possible templates involving edges that cross the line separator. This gives an instance on a smaller

rectangle that was already solved as part of the dynamic programming, and the optimum solution is in the lookup table. After looking at the optimum solutions for subproblems defined using all possible line separators and templates, we use the line separator and the template that minimizes the cost, and store that in the lookup table.

Solving the instance arbitrarily in the base case can only affect the cost by $(n-1) \times L/n^2 < L/n < \frac{\epsilon}{3}OPT$, since the tree has only $n-1$ edges. The cost of the output of the algorithm is then within $\frac{\epsilon}{3}OPT$ of the cost of the optimum r-friendly degree-restricted spanning tree. Since the set of r-friendly degree-restricted spanning trees is a superset of the set of (m,r)-light degree-restricted spanning trees, the cost of the output is within a factor of $(1+\epsilon/3)$ of the cost of the optimum (m,r)-light degree-restricted spanning tree. It follows from the Structure Theorem, with $\epsilon/3$ as the approximation factor, that the output of the algorithm has cost at most $(1+\epsilon)OPT$.

It may seem that the Structure Theorem is more general than necessary, since it requires the degree-restricted spanning tree to be portal respecting. Unlike previous approximation schemes for geometric problems, this restriction does not improve the asymptotic efficiency of the dynamic programming algorithm. Interestingly, as the reader will find in the next section, the portal respecting property is essential for the proof of the Structure Theorem, specifically for the proof of Lemma 12.

2.2 Proof of Structure Theorem

The theorem is of course similar to known results for other geometric problems. The proof is also somewhat similar: we start with an optimum solution and whenever it crosses the tiling too often, we modify it so that it doesn't. Difficulties arise in tracking the cost increases because the degree bound precludes many solutions. To reason about cost increases, we will use some fixed optimum solution, say T_0, rooted at an arbitrary node. We let d denote the degree bound. We will use the following objects extensively.

DEFINITION 2 (d-FOREST) *A* d-forest *in the instance is a collection of pairs* $(T_1,u_1),\ldots,(T_k,u_k)$ *where the* T_i*'s are node-disjoint trees of degree at most* d *which together contain all nodes, and* $u_i \in T_i$ *is the* representative node *for* T_i*. We require that* u_1 *has degree at most* $d-1$ *and* $u_2,u_3,\ldots,u_k$ *have degree at most* $d-2$*. We call* u_1 *the* start node.

DEFINITION 3 (REARRANGEABLE PATH) *If* T *is any degree-restricted spanning tree, a* free node *is a node with degree at most* $d-1$*. For any tree edge* e*, the* rearrangeable path of e with respect to T *is the path* p *from* e *to a descendant free node such that* p *has the least number of edges of all such paths. If there are multiple paths with this number of edges, pick the rightmost path.*

REMARK 1 A rearrangeable path has at most $\lceil \log_2 n \rceil$ edges, since otherwise the subtree rooted at this edge would have degree 3 for depth at least $\log_2 n$,

and thus more than n nodes (our convention is to include e in the path.) If two tree edges have rearrangeable paths that contain a common edge, then one must be a descendent of the other, and hence one path is contained in the other.

Suppose T is a degree d tree rooted at v_0 and $e = (v_1, v_2)$ is some edge whose rearrangeable path is $(v_1, v_2, \ldots, v_l)$, where v_l is a free node. Deleting every edge on this path partitions the nodes of T into subtrees $T_1, T_2, \ldots, T_l$, where v_j lies in T_j. Furthermore, each of $v_2, v_3, \ldots, v_l$ now has degree at most $d-2$ (since we deleted two edges incident to each $v_1, \ldots, v_{l-1}$, and one edge incident to v_l) and v_1 has degree at most $d-1$. Thus $(T_1, v_1), (T_2, v_2), \ldots, (T_l, v_l)$ is a d- forest, where each v_i for $i \geq 2$ is both the root and the representative node for T_i.

Now suppose we iterate the process and remove a second rearrangeable path from the d-forest. Let $e' = (u_1, u_2)$ be some edge whose rearrangeable path is $(u_1, u_2, \ldots, u_m)$, where u_m is a free node. Note that any path in the d-forest must be contained in a single one of the subtrees; assume without loss of generality that e''s rearrangeable path is contained in T_l. Removing all edges in the rearrangeable path of e' partitions the nodes of T_l into subtrees $T_l^1, \ldots, T_l^m$. By reasoning similar to the above, we know that each of $u_2, u_3, \ldots, u_m$ now has degree $d-2$. Node u_1 has degree $d-1$, but we do not take u_1 to be the representative node of T_l^1, since $v_l \in T_l^1$ is still its representative node. It follows that $(T_1, v_1), (T_2, v_2), \ldots, (T_{l-1}, v_{l-1}), (T_l^1, v_l), (T_l^2, u_2), (T_l^3, u_3), \ldots (T_l^m, u_m)$ is a d-forest, where each v_i and u_i for $i \geq 2$ is both the root and representative node of its respective subtree. This argument can be applied inductively to show that removing any number of rearrangeable paths results in a d-forest.

Now we are ready to prove a crucial property of the optimum tree. The following lemma relies on the fact that the shortest salesman path on k nodes in a square of sidelength M has length $O(\sqrt{k}M)$. The well-known *salesman strip tour* is a construction that achieves this bound (See Lemma 15 at the end of this section for a more general version.) We will use it often.

LEMMA 4 (CROSSING LEMMA FOR DRMST) *For any constant c there is a constant $c' > 0$ such that the following is true for any optimum degree-restricted minimum spanning tree T_0. Let S be any straight line segment of length $s \geq 1$. Then T_0 has at most $c' \log^5 n$ edges that cross S and have rearrangeable paths longer than $s/(c \log n)$.*

PROOF: Assume for contradiction's sake that the number of such edges exceeds $c' \log^5 n$. We modify the tree to produce one of lower cost, contradicting optimality.

Recall that the bounding box has length $L = n^3$, so rearrangeable paths have length at most $O(n^3 \log n)$. Partition the edges with rearrangeable paths of length greater than $s/c \log n$ into at most $N \leq 4 \log n$ different categories, $C_0, C_1, C_2, \ldots, C_N$ where C_j contains edges whose rearrangeable paths have length from $s2^j/c \log n$ to $s2^{j+1}/c \log n$. The pigeonhole principle implies that some category, say C_i, contains $k' > \frac{c'}{4} \log^4 n$ crossing edges. Since a rearrangeable path contains at most $\log n$ edges and two paths intersect iff one lies inside the other, we can pick $k = k'/\log n$ paths from this category that are pairwise

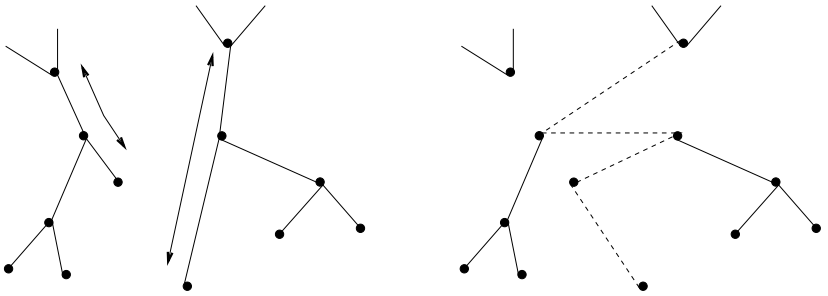

Figure 3: On the left, the tree before a patching action The arrows indicate rearrangeable paths. On the right, the tree after removing these paths and reconnecting with a salesman path.

edge-disjoint. Note that the paths have lengths between $M/2$ and M, where $M = s2^{j+1}/c \log n$.

Remove all edges in all k paths; this gives a d-forest whose cost is lower by at least $kM/2$. Since each path removal gives rise to at most $\log n$ subtrees, the d-forest has at most $k \log n + 1$ subtrees.

Connect the representative nodes of the at most $k \log n + 1$ trees with the shortest salesman path, whose length is at most $O(\sqrt{k \log n + 1}(2M + s))$. Since $k \geq \frac{c'}{4} \log^3 n$ and $M > 2s/c \log n$ the total added cost is $O(\sqrt{c'} \log^2 n(M + s))$, which for large enough c' is lower than the cost $kM/2$ saved earlier when we removed edges. Thus we have lowered total cost, while keeping all degrees at most d. □

The proof of the Structure Theorem uses reasoning similar to Lemma 4. We start with the optimum degree-restricted spanning tree T_0 and repeatedly remove rearrangeable paths from a degree-restricted spanning tree. The result of removing these rearrangeable paths is a d-forest; denote its representative nodes by $v_1, v_2, \ldots, v_k$, where v_1 is the start node. We then add a single salesman path starting at v_1 and visiting all of $v_2, v_3, \ldots, v_k$ (not necessarily in this order), which gives a degree-restricted spanning tree again. We refer to this process of reconnecting a d-forest with a salesman path as a *patching action*. See Figure 3.

The power of this approach comes from the fact that *any* salesman path suffices, and that Arora's Structure Theorem for TSP shows the existence of near-optimum salesman paths that cross the tiling boundaries "not too often." As we will show, the optimum tree can be gradually modified using iterated patching actions to be (m, r)-light.

However, this iterative approach has a complication: even after one patching action, the modified tree is no longer optimum and hence Lemma 4 cannot be applied to it. Intuitively, this problem is resolved by noting that the patching actions leave most of the original tree untouched. This motivates the next few definitions and lemmas.

DEFINITION 5 (CORE EDGE AND CORE REARRANGEABLE PATH) *Let T be some tree of degree at most d and T_0 be the optimum tree. Let e be an edge in $T \cap T_0$ and p be its rearrangeable path with respect to T_0. We say that e is a* core edge *of T if the set $p \cap T$ forms a path in T, and that path ends at a free node of T. If e is a core edge of T, then the path $p \cap T$ is called its* core rearrangeable path *with respect to T.*

REMARK 2 Core rearrangeable paths with respect to T are subpaths of the original rearrangeable paths (i.e. paths that were rearrangeable paths with respect to T_0.) Hence they satisfy many of the same properties. In particular, two core rearrangeable paths intersect if and only if one path is contained in the other, and also the length of a core rearrangeable path with respect to T is at most $\log_2 n$.

DEFINITION 6 (CORE) *Let T be some tree of degree at most d and T_0 be the optimum tree. A* core *of T is a set of edges $C \subseteq T_0 \cap T$ such that every edge e in C is a core edge and its core rearrangeable path with respect to T also lies in C.*

REMARK 3 A core of a tree is not unique, but there is a largest core that contains all other cores. In order to see that such a core is well-defined, suppose we have two cores A and B. We want to show that $A \cup B$ is a core. Suppose $e \in A \cup B$ is an edge, then $e \in A$ or $e \in B$, which implies that e's core rearrangeable path is in A or B, and is thus in $A \cup B$. Hence $A \cup B$ is a core. This motivates the definition of the following well-defined object:

DEFINITION 7 *Let T be some tree of degree at most d and T_0 be the optimum tree. We denote by $C^*(T)$ the largest core in T.*

If T is derived from successive modifications of a tree T_0, a core of T can be intuitively thought of as the remnant of T_0 in T, with the original rearrangeable paths "roughly intact."

The next corollary explains our interest in cores; its proof is nearly identical to the proof of Lemma 4.

COROLLARY 8 (TO LEMMA 4) *For any constant c there is a constant $c' > 0$ such that the following is true for a core C of any degree-restricted spanning tree. Let S be any straight line segment of length s. Then C has at most $c' \log^5 n$ edges that cross S and have core rearrangeable paths longer than $s/(c \log n)$.*

REMARK 4 Patching actions can now be executed on a tree T in the following manner. We remove a set of core rearrangeable paths from T. The result of removing these paths from T is a d-forest, because core rearrangeable paths terminate at free nodes. The d forest's subtrees can then be connected together with a salesman path.

Now consider what happens to $C^*(T)$ after a patching action on T that results in a tree T'. What is the largest core $C^*(T')$ of the modified tree? Lemma 11 answers this question. First, we need two definitions.

DEFINITION 9 (UPSTREAM EDGE) *If T is a degree-restricted spanning tree and u is a node, then a core edge e in T whose core rearrangeable path passes through u is called an* upstream edge associated with u, T.

REMARK 5 The set of all upstream edges associated with node u forms a path. Consider two edges whose core rearrangeable paths p_1 and p_2 pass through u; either p_1 is contained in p_2 or vice versa. Thus, there exists a largest path that contains all such upstream edges. Such a path contains at most $\log_2 n$ edges, since all core rearrangeable paths have at most $\log_2 n$ edges. One can think of this path as the path of all nodes that are "upstream" from the node u. These observations lead to the following defintion:

DEFINITION 10 (UPSTREAM PATH) *If T is a degree-restricted spanning tree and u is a node, then the path of edges that are upstream with respect to u, T is called the* upstream path associated with u, T.

The next Lemma shows that patching actions on core rearrangeable paths do not greatly affect the core (as noted, the upstream path has length at most $\log_2 n$.)

LEMMA 11 *Suppose tree T' results from a patching action on tree T that involved the removal of core rearrangeable paths $p_1, \ldots, p_k$ and the addition of a salesman path p that began at node u. If f is the upstream path associated with u, T, then*

$$C^*(T') \supseteq C^*(T) \setminus (f \cup \cup_i p_i).$$

PROOF: Let $C' = C^*(T) \setminus (f \cup \cup_i p_i)$. Let $e \in C'$, and $p_{e,T}$ be e's core rearrangeable path with respect to T.

Suppose that $p_{e,T}$ is not completely contained in $C^*(T')$. The only way this is possible is if an edge in $p_{e,T}$ intersected one of the removed core rearrangeable paths, say p_i. Since core rearrangeable paths intersect if and only if one is contained in the other, this implies that p_i is contained in $p_{e,T}$. If $p_i = (u_1, \ldots, u_k)$, then after the patching action, u_1 is a free node (u_1 is not the start node of p, since $e \notin f$.) Thus $p_{e,T'} = T' \cap p_{e,T}$ is a path that terminates at the free node u_1 and is hence a core rearrangeable path. □

Call the modification of $C^*(T)$ to $C^*(T')$ after a patching action a *core update*. Now we are ready to prove the main theorem.

PROOF: (Of Structure Theorem) The proof consists of describing a procedure that, over $N = O(\log n)$ phases, produces a $1/3 : 2/3$-tiling of the bounding box, while simultaneously converting an optimum degree-restricted spanning tree, T_0, into a degree-restricted spanning tree T_N that is (m, r)-light with respect to this tiling. The i'th phase refines the depth i tiling into a depth $i + 1$ tiling, and

transforms T_i into T_{i+1}. T_{i+1} is (m, r_{i+1})-light with respect to the refined tiling, where $r_{i+1} = O(\log^5 n)$, and it satisfies the degree constraints. At each phase we update the core of the tree as described earlier.

INVARIANTS: We maintain the following four invariants:

(a) The tree T_{i+1} consists of three types of edges: the core $C^*(T_{i+1})$, a set of salesman paths $p_1, \ldots, p_j$ (these are edge-disjoint but need not be vertex-disjoint) that became part of the tree during patching actions in previous iterations, and a set of *fixed edges* $\mathcal{F}_i \subseteq T_0$ that will not be removed from the tree in subsequent phases. The set $\mathcal{F}_i$ is a union of some paths $f_1, \ldots, f_j$ that were upstream paths consisting of edges from T_0 whose core rearrangeable paths were found to intersect the start node of the salesman paths created in an earlier phase (Definition 10.) We have

$$T_{i+1} = C^*(T_{i+1}) \cup (\cup p_j) \cup (\cup f_j)$$

As noted in the remarks preceding Definition 10, each upstream path f_j has only $\log_2 n$ edges; these edges may also be contained in the core, but such cautious overaccounting will not hurt our result (for the purposes of the proof, they are treated as core edges.)

(b) Each level $i+1$ rectangle contains edges from at most $i+1$ of the aforementioned salesman paths.

(c) T_{i+1} crosses each level $i+1$ line separator at most $r_{i+1} = O(\log^5 n)$ times.

(d) T_{i+1} is portal-respecting, where the portal parameter m is as quantified below.

Note that invariants (c) and (d) imply that T_{i+1} is (m, r_{i+1})-light.

SPECIFICATION OF PARAMETERS: Now we specify the values of the various parameters. We restrict the portal parameter m to satisfy $m > 36 \log n$ and $m = O(\log n/\epsilon)$, the constant c (it may depend on ϵ) to satisfy $c > 2m/\log n$, and set t to be the bound from Corollary 8 on the number of edges with core rearrangeable paths of length at least $s/c \log n$ that cross a line segment of length s. Recall that $t = O(\log^5 n)$, and that t depends only on c and not on s.

We will prove that the costs of the trees satisfy the recurrence

$$\text{cost}(T_{i+1}) \leq \text{cost}(T_i) + \frac{11\text{cost}(T_i)}{m}.$$

After all $N = O(\log n)$ phases, this implies that $\text{cost}(T_N) \leq (1 + \frac{11}{m})^N OPT \leq (1+\epsilon)OPT$, since $m = O(\log n/\epsilon)$.

DETAILS OF PROCEDURE: Phase i considers each level i rectangle of the current tiling one by one, divides the rectangle into two by picking a suitable line separator, and modifies the tree in such a way that it still maintains our four invariants with respect to the new line separator.

Let R be a level i rectangle whose longer side has length W. Let $T_i|_R$ denote the portion of T_i that lies in R. For each action of Phase i, we first describe the modifications, then show how the actions together maintain the invariants, and lastly bound the cost of T_{i+1}.

PHASE i'S ACTION IN R, FIRST STEP: Suppose we choose a line separator uniformly at random from the set of line separators in the middle third of R. Lemma 14 implies that the expected number of times this line separator is crossed by the edges of T_i is at most $C = 3\text{cost}(T_i|_R)/W$. If $C < t$, we do not modify $T_i|_R$ at all, and just pick such a line separator to get two level $i+1$ rectangles.

On the other hand, suppose $C \geq t$. Regardless of which line separator we pick, we may need to modify the tree in order to reduce the number of edges that cross the line separator. Let CE be the set of edges that cross our chosen line separator. We partition the edges of CE into six categories:

$$CE = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6,$$

where the categories are defined below:

1. E_1 consists of crossing edges belonging to the i salesman paths in R.
2. E_2 consists of crossing edges belonging to the core, whose core rearrangeable paths are shorter than $W/c\log n$ and are completely contained in R.
3. E_3 consists of crossing edges belonging to the core, whose core rearrangeable paths are longer than $W/c\log n$.
4. E_4 consists of crossing fixed edges e_f whose rearrangeable paths in T_0 are of length greater than $W/c\log n$.
5. E_5 consists of crossing fixed edges whose rearrangeable paths in T_0 are of length less than $W/c\log n$ and are completely contained in R.
6. E_6 consists of crossing fixed or core edges whose rearrangeable paths in T_0 and core rearrangeable paths in T_i, respectively, are of length less than $W/c\log n$, but are not contained in R. i.e. their rearrangeable paths each contain at least one edge with an endpoint outside of R.

Step 1 proceeds by first choosing a line separator S for rectangle R and then modifying the tree so that the total number of edges crossing S is small and invariants (a), (b), and (c) are maintained.

The following six claims show how to bound the number of edges in each successive category, or how to locally modify the tree —namely, leaving it unchanged outside rectangle R— in order to reduce the crossings in a way that is consistent with invariants (a), (b), and ultimately (c). It is important to note that the following bounds/modifications on E_1, E_2, E_3, E_4 and E_5 do not depend on which line separator is chosen.

CLAIM 1 (MODIFICATION): *Given any line separator, the tree can be modified so that the number of crossing edges in E_1 is reduced to $2i$. The modification preserves invariants (a) and (b), and results in a cost increase of at most $O(iW)$.*
PROOF: We can break these edges and then apply Lemma 13 at most i times to reconnect the freed nodes, thus raising cost by at most $O(iW)$, and reducing the number of crossings from these salesman paths to $2i$. Clearly, the added edges are completely contained in R. □

CLAIM 2 (MODIFICATION): *Given any line separator, the tree can be modified so that all edges in E_2 are removed and replaced with a salesman path that crosses the line separator at most twice. This preserves invariants (a) and (b).*
PROOF: We remove all the edges of E_2 along with their core rearrangeable paths, thus obtaining a d-forest F. We then execute a patching action on F, but use a salesman path p that crosses S at most twice (Lemma 13 shows such a path exists), and update the core. The upstream path associated with the start node of p now becomes a set of fixed edges. Clearly, invariants (a) and (b) are maintained since the added salesman path lies entirely in R. □

CLAIM 3 (BOUND): *For any line separator, E_3 contains at most t crossing edges.*
PROOF: Apply Corollary 8. □

CLAIM 4 (BOUND): *For any line separator, E_4 contains at most t edges.*
PROOF: Since each edge in E_4 corresponds to a unique edge in T_0 that crosses S with long rearrangeable path, there are at most t of these by Lemma 4. □

CLAIM 5 (BOUND): *For any line separator, E_5 contains at most $i \log_2 n$ edges.*
PROOF: A fixed edge $e \in E_5$ must be associated with some salesman path that lies partly in R. Since at most $\log_2 n$ upstream edges (and hence $\log_2 n$ fixed edges) are associated with each salesman path and by invariant (b) there are at most i salesman paths in R, the total number of edges in E_5 cannot exceed $i \log_2 n$. □

Finding a scheme to bound the number of edges in E_6 is much trickier. We cannot proceed with a modification as we did in Claim 2 for the following reason. Recall that the edges in E_6 consist of core edges and fixed edges whose rearrangeable paths with respect to T_0 and core rearrangeable paths with respect to T_i, respectively, are not contained in the rectangle R. Therefore, attempting to replace these core rearrangeable paths with a salesman path will modify the tree in neighboring rectangles and thus would potentially violate invariant (b) for these neighboring rectangles.

We do not modify the tree in order to remove edges in E_6, but rather prove the existence of a line separator S such that the number of crossing edges of type E_6 is small. Fortunately, all the modifications and bounds we proved for the sets E_1, E_2, E_3, E_4 and E_5 are independent of the chosen line separator and thus we are free to choose any S we wish, without fear of invalidating these other bounds.

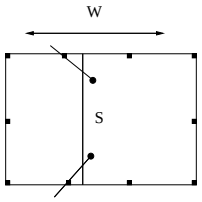

Figure 4: Crossing edges of type E_6.

CLAIM 6 (BOUND): *There exists a line separator S of R such that the number of edges in category E_6 is at most $r_i/2$.*
PROOF: The following lemma is similar to Lemma 14.

LEMMA 12 *For a line separator chosen uniformly at random from the set of line separators, the expected number of crossing edges in E_6 is at most $r_i/6$.*

PROOF: Recall that invariant (d) guarantees that the tree T_i is portal respecting. We had already set the portal parameter m to satisfy $m > 36 \log n$. By invariant (c), there are at most r_i edges of T_i that cross into R from any one of the sides of R. We have chosen constant c so that $c \log n > 2m$. This implies that along the longer side of R, the distance between any two portals is at least $2W/c \log n$ (recall that W is the length of the longer side of R.)

Fix some line separator S. Recall that line separators are placed in the middle third of the rectangle only. Since the interportal distance is at least $2W/c \log n$, there are only two portals within distance $W/c \log n$ of S. If S is vertical (horizontal), one of these portals is above (to the left of) S and the other is below (to the right of) S. See Figure 4. Since edges that cross S in E_6 must be derived from rearrangeable paths/core rearrangeable paths of length at most $W/c \log n$, it follows that such paths may enter R only through these two portals. By averaging, we know that the expected number of edges that enter a given portal in the middle third of the rectangle is at most $r_i/(m/3) \leq r_i/12 \log n$. By linearity of expectations, it follows that the number of edges that cross into the two portals closest to a line separator chosen uniformly at random from the middle third of R is at most $r_i/6 \log n$. Since each of these paths can induce at most $\log_2 n$ crossings in E_6, the expected number of such crossings is at most $r_i/6$. □

Let $\mu = 3C$. If we pick a line separator uniformly at random from the set of line separators in the middle third of R, with probability at least $2/3$, the number of total crossings is at most μ by Lemma 14, and with probability at least $2/3$, the number of crossings edges in E_6 is at most $r_i/2$. Therefore a line separator S exists for which both events occur, and we choose S to divide the rectangle. □

We have thus shown how to modify the tree in order to bound each type of edge, and that the modifications preserve invariants (a) and (b). We now show

that these modifications preserve invariant (c) —namely, the modified tree does not cross S more than $O(\log^5 n)$ times.

BOUNDING THE TOTAL NUMBER OF EDGES CROSSING S: We have modified the tree in order to bound each type of crossing edge: E_1, E_2, E_3, E_4, E_5 and E_6. In order to see that invariant (c) is maintained, note that the total number of times the modified tree crosses S is at most $r_{i+1} = 2+2i+3t+i\log_2 n+r_i/2$. Since $r_1 \leq 4t$, this recurrence relation solves to $r_{i+1} \leq 4t\sum_{0\leq j\leq i+1} 1/2^j \leq 8t = O(\log^5 n)\forall i$.

BOUNDING THE COST INCREASE: As proved in the discussion of Claim 1, the cost of removing edges in E_1 is at most $O(iW) = WO(\log n)$. For n sufficiently large, this cost is $< W\mu/9m < \text{cost}(T_i|_R)/m$. As proved in the discussion of Claim 2, the cost of removing edges in E_2 is the cost of the salesman path that visited at most $\mu \cdot \log_2 n$ nodes. We know such a salesman path costs at most $O(W\sqrt{\mu \log_2 n + 1})$. For n sufficiently large, this cost is $< W\mu/9m \leq \text{cost}(T_i|_R)/m$.

PHASE i'S ACTION ON R, SECOND STEP: The second step of Phase i modifies the tree so that the result is portal respecting (invariant (d).) The step involves moving all $\leq 3C$ crossings to portals. This action is accomplished by adding two vertical detours of length at most $W/2(m-1)$. Thus at the end all crossings are portal respecting. For n sufficiently large, the total cost of moving crossings to portals is then at most $3CW/(m-1) \leq 9\text{cost}(T_i|_R)/(m-1)$.

Call the tree resulting from Phase i Steps 1 and 2 on all level i rectangles T_{i+1}. We have shown that T_{i+1} satisfies all four invariants, and that $\text{cost}(T_{i+1})$ satisfies the recurrence relation $\text{cost}(T_{i+1}) \leq \text{cost}(T_i) + \frac{11\text{cost}(T_i)}{m}$. Furthermore, each T_i for $i = O(\log n)$ is (m, r)-light, where $r = r_N = O(t)$. □

We state two Lemmas that were used above.

LEMMA 13 (PATCHING LEMMA FOR TSP [1, 2]) *Let S be any line segment of length s and P be any salesman path that crosses S at least three times. Then we can break the path in all but two of these places, and add to it line segments lying on S of total length at most $3s$ such that P changes into a salesman path P' that crosses S at most twice.*

LEMMA 14 ([1]) *Let $\mathcal{D}$ be any collection of straight line segments whose total length is C, and which lie entirely inside a rectangle of size W. Then the expected number of segments of $\mathcal{D}$ that cross a line separator chosen uniformly at random from the middle third of the rectangle is at most $3C/W$.*

PROOF: The line separator is picked randomly from the middle third of the rectangle, a region of length at least $W/3$. The probability that a line segment of length l crosses this separator is at most $3l/W$; the Lemma follows by linearity of expectations. □

The next two lemmas are standard [6]. We cite them more generally in the context of $\mathbb{R}^k$; we will need these stronger lemmas when we generalize the Structure Theorem to $\mathbb{R}^k$.

LEMMA 15 (SALESMAN STRIP TOUR) *Given j points in $\mathbb{R}^k$ contained in a k-cell with maximal side length s, there exists a salesman tour of these j points that costs $O(sj^{1-1/k})$.*

COROLLARY 16 (SMART SALESMAN STRIP TOUR) *Given j points in $\mathbb{R}^k$ contained in a k-cell with maximal side length s and a hyperplane separator S, there exists a salesman tour of these j points that costs $O(sj^{1-1/k})$ and that crosses the hyperplane separator at most twice.*

PROOF: Visit the nodes in the k-cells on either side of the hyperplane separator with a separate salesman strip tour. Break one edge from each strip tour and connect the loose ends to form one salesman tour that crosses the hyperplane separator twice. The cost of this strip tour is $2O(sj^{1-1/k}) + 2s = O(sj^{1-1/k})$. □

2.3 Structure Theorem in $\mathbb{R}^k$

An analogous result holds for $\mathbb{R}^k$. Our 1/3:2/3-tiling in $\mathbb{R}^k$ is the obvious generalization of the tiling in $\mathbb{R}^2$, consisting of k-dimensional boxes in lieu of rectangles and $k-1$ dimensional hyperplane separators in lieu of line separators. The boundary of such a k-dimensional box consists of $2k$ different $k-1$-dimensional hyperplanes. The portal parameter m specifies the number of portals that occur on each boundary hyperplane; these portals are placed regularly on an orthogonal lattice on the hyperplane. If W is the length of the longest side of the hyperplane, the distance between any two adjacent portals is between $W/m^{1/(k-1)}$ and $W/(3m^{1/(k-1)})$.

We then have the following, more general form of Theorem 1.

THEOREM 17 (STRUCTURE THEOREM FOR DRMST IN $\mathbb{R}^k$) *For $d > 2$, there exists a $1/3 : 2/3$-tiling and a degree-restricted spanning tree with cost at most $(1+\epsilon)OPT$, such that the tree is (m, r)-light with respect to this 1/3:2/3-tiling. Here $m = O((\log n/\epsilon)^{k-1})$ and $r = O(\log^{2k+1} n)$.*

The proof of the Structure Theorem in $\mathbb{R}^k$ is analogous to the proof for the Structure Theorem in the plane. There are only two fundamental changes that need to be made. First, the upper bound on the cost of a strip tour that visits j nodes in a k-dimensional box of length s is $O(sj^{1-1/k})$ (see Lemma 15.) Second, the Patching Lemma for TSP (Lemma 13) does not hold for hyperplane separators. We do not repeat the entire treatment in Section 2.2, but rather just present the necessary recalculations induced by these and a few other considerations.

MODIFICATIONS TO LEMMA 4: Let S be a $k-1$ dimensional hyperplane with longest side of length s. In $\mathbb{R}^k$, an optimum tree has at most $c' \log^{2k+1} n$ edges that cross S and have rearrangeable paths of length greater than $s/c \log n$, for some $c' > 0$. The proof of this fact is the same as in Lemma 4, but with the following changes:

- We assume that the number of edges crossing S is at least $t = c' \log^{2k+1} n$.

- Then, there is some C_i with more than $\frac{c'}{4}\log^{2k} n$ edges such that the lengths of the rearrangeable paths of these edges are at least $s2^i/c\log n$. Thus, we can find at least $l = \frac{c'}{4}\log^{2k-1} n$ crossing edges in C_i whose rearrangeable paths are pairwise disjoint.

- If we remove these l rearrangeable paths, we reduce the cost by $lM/2$. In $\mathbb{R}^k$, the salesman path that visits $l\log n + 1$ nodes will cost at most $O((l\log n + 1)^{1-1/k}(2M + s))$, from Lemma 15.

- Since $l \geq \frac{c'}{4}\log^{2k-1} n$, this cost increase is $O(\sqrt{c'}\log^{2k-2} n(M + s)$, which for large enough c' is lower than $lM/2$.

MODIFICATIONS TO COROLLARY 8: The corollary changes with the corresponding change in Lemma 4. Let S be any $k-1$ dimensional hyperplane with longest side of length s. In $\mathbb{R}^k$, a core has at most $c'\log^{2k+1} n$ edges that cross S and have core rearrangeable paths of length greater than $s/c\log n$, for some $c' > 0$.

MODIFICATIONS TO PROOF OF STRUCTURE THEOREM: The proof of the Structure Theorem in $\mathbb{R}^k$ is the same as the proof in $\mathbb{R}^2$, but with the following recalculations of upper bounds on costs and crossing edges:

- We keep the same four invariants, except for a small change in invariant (c). In $\mathbb{R}^k$, the number of times T_{i+1} crosses a level $i+1$ hyperplane separator is $r_{i+1} = O(\log^{2k+1} n)$.

- The portal parameter m satisifes $m > (36(k-1))^{k-1}\log^{k-1} n$ and $m = O((\log n/\epsilon)^{k-1})$. The constant c satisfies $1/(c\log n) \leq 1/(9m^{1/(k-1)})$, and the bound t from the Crossing Lemma satisfies $t = O(\log^{2k+1} n)$.

- For Phase i, Step 1, we had categorized the edges crossing the hyperplane separator S as

$$CE = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6.$$

- *Modifications to Claim 1:* The total number of edges that cross our hyperplane separator is at most $\mu = 3C$. Thus each salesman path can cross S at most μ times. For each salesman path, these crossings can be reduced to at most two by augmenting the salesman path with line segments contained in the hyperplane separator. The aggregate cost of these line segments is $O(iW\mu^{1-1/(k-1)})$. The veracity of this approach can be established by generalizing the proof of the Patching Lemma (Lemma 3) in [2].

- *Modifications to Claims 3 and 4*: The same arguments as before follow; the number of crossing edges in E_3 and E_4 is at most $t = c'\log^{2k+1} n$.

- *Modifications to Claim 6:* Again, we pick a hyperplane separator so that the number of edges in E_6 is at most $r_i/2$. The proof that such a hyperplane separator exists relies on a modified version of Lemma 12.

  MODIFICATIONS TO PROOF OF LEMMA 12: The expected number of edges in E_6 that cross a hyperplane separator chosen uniformly at random from the middle third of the box is still $r_i/6$. The recalculations are as follows. We had already set $m = b\log^{k-1} n$ where $b \geq (36(k-1))^{k-1}$ and c such that $W/c\log n$ is smaller than half the interportal distance on any side of the box. Thus, the edges in E_6 can have rearrangeable paths/core rearrangeable paths that originate from at most $2(k-1)b^{(k-2)/(k-1)}\log^{k-2} n$ portals. The average number of paths that enter a given portal in the middle third of some side of the box is at most $3r_i/(b\log^{k-1} n)$ (since $b\log^{k-1} n/3$ is the total number of portals on the middle third of the hyperplane.) Thus, by linearity of expectations, the expected number of edges that pass through the $2(k-1)b^{(k-2)/(k-1)}\log^{k-2} n$ portals closest to a randomly chosen hyperplane separator is $6(k-1)r_i b^{-1/(k-1)}/\log n \leq r_i/6\log n$. Since each of these paths can induce at most $\log_2 n$ crossing edges, the expected number of edges in E_6 is at most $r_i/6$.

  We can then pick a hyperplane separator that is crossed by at most $\mu = 3C > t$ edges, of which at most $r_i/2$ are in E_6.

- *Total number of edges crossing a hyperplane separator:* The recurrence relation for the number of times T_i crosses S is $r_{i+1} = 2+2i+3t+i\log_2 n + r_i/2$, which implies that $r_i \leq 8t = O((\log n)^{2k+1})$.

- *Total cost increase due to modifications in Step 1:* We now bound the cost increase due to modifying the tree in order to eliminate edges in E_1 and E_2. Let $c'' = 36/\epsilon$. Removing the crossing edges in E_1 involves patching the $i = O(\log n)$ salesman tours, which will cost at most $O(iW\mu^{1-1/(k-1)}) \leq W\mu/c''\log n$, for n sufficiently large. Modifications for eliminating crossings in E_2 involve a salesman path that costs at most $O(W(\mu\log n + 2)^{1-1/k}) \leq W\mu/c''\log n$, for n sufficiently large. Thus the cost increase due to Phase i, Action 1 is at most $2W\mu/\log n = \epsilon\text{cost}(T_i|_R)/2\log n$.

- *Total cost increase after Step 2:* The cost increase due to Phase i, Action 2 is the cost of detouring each edge by at most most $W/m^{1/(k-1)}$. For $m = O((\log n/\epsilon)^{k-1})$, Step 2 will cost at most $\epsilon\text{cost}(T_i|_R)/2\log n$. Thus, the total cost is $\text{cost}(T_{i+1}) = (1+\epsilon/\log n)^{i+1}OPT$.

3 Red-Blue Separation

The proof of the existence of a PTAS for RBSP follows the proof of the existence of a PTAS for TSP in [2] almost exactly. Apart from a few minor modifications,

the only new result that needs to be proved is a Patching Lemma for RBSP, to be used in lieu of the Patching Lemma for TSP (Lemma 3 of [2].) This gives an algorithm running in $n(\log n)^{O(1/\epsilon)}$ time.

The PTASs for RBSP and TSP are similar in nature to the QPTAS for DRMST presented above. We outline the steps below while simultaneously pointing the reader to the appropriate section of [2] for a more complete treatment, and then prove our new result, the Patching Lemma for RBSP.

Our PTAS for RBSP calculates a simple polygon containing all red points and no blue points, which we will call a *separating polygon*, with weight at most $(1+\epsilon)$OPT (where OPT is the weight of the optimum separating polygon), in nearly linear time. The algorithm consists of three main steps that parallel the three steps of the PTAS for TSP, found in Section 2.1.1 of [2].

STEP 1: PERTURBATION. The perturbation algorithm is a "preprocessing" step that slightly transforms the instance in order to enforce three useful conditions: (i) all nodes have integral coordinates, (ii) each (nonzero) internode distance is at least 8, (iii) the maximum internode distance is $O(n)$. An instance that satisfies these three properties is described as *well-rounded*.

The perturbation for TSP is described in Step 1 of Section 2.1.1 in [2]. The perturbation algorithm places a grid of sufficiently small granularity on the instance and moves each node to the closest grid point. Note that it is possible for points of different colors to be moved to the same grid point. The perturbation of a TSP instance is described in detail in [2], where the "sufficiently small granularity" is presented, and where it is proved that the optimum salesman path on the perturbed instance is only neglibly more costly than the optimum salesman path on the original instance.

The perturbation for TSP generalizes to RBSP, except for one minor modification. The original TSP perturbation makes the assumption that $OPT \geq L$, where OPT is the cost of the optimum salesman tour and L is the largest internode distance. In general, this is not the case for RBSP. We do know, however, that the length of the perimeter of a separating polygon exceeds the maximum distance between red nodes; call this internode distance L_R. If we define the *red bounding box* as a square of side length L_R that contains all red points, then clearly a separating polygon with minimum perimeter that encloses red points is completely contained in the red bounding box. When computing the minimum perimeter separating polygon, we can ignore all blue points that are not contained in the red bounding box.

Thus, our perturbation is exactly like the TSP perturbation, except we only consider the red bounding box of length L_R.

STEP 2: CONSTRUCTING A SHIFTED QUADTREE. We first define a recursive partitioning of the red bounding box, called a *dissection*. A dissection is a 4-ary tree consisting of squares (not rectangles, as in the 1/3:2/3-tiling), whose root is the bounding box. Each square is recursively partitioned into four squares of equal size, which are its children. The partitioning stops at squares that contain at most one node (or multiple nodes with internode distance 0), which are the leaves of the 4-ary tree. See Figure 5. Since the perturbation ensures us that

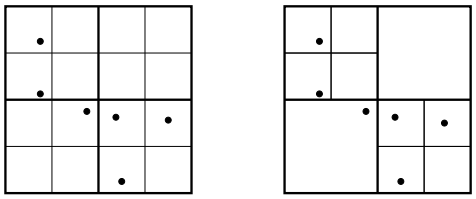

Figure 5: On the left, the dissection of an instance. On the right, the corresponding quadtree.

the red bounding box is of length $O(n)$ and the internode distance is at least 8, the depth of the dissection is at most $O(\log n)$ and contains at most $O(n^2)$ squares.

A dissection can be randomized, as described in Section 2.1 of [2], by the choice of two random integers $0 \leq a, b < L_R$, to create a *dissection with shift* (a, b).

A *quadtree* is similar to a dissection, except that the partitioning ceases at squares that contain at most one node, which are the leaves of the 4-ary tree. A quadtree contains $O(n \log n)$ squares. Similarly, a quadtree can be randomized to create a *quadtree with shift* (a, b).

On each edge of a quadtree or dissection square, we place a set of m equally spaced portals. We also place a portal at each of the four corners of the square. A separating polygon is (m, r)-light if it crosses into any given square at most r times, and each time at one of its $4m + 4$ portals.

Thus, Step 2 constructs a quadtree with shift (a, b), where $0 \leq a, b < L_R$ are random integers. This can be performed in $O(n \log^2 n)$ time with a sorting-based algorithm.

With this notion of a randomized quadtree, we can state our structure theorem, analogous to Theorem 2 in [2] and Theorem 1 above:

THEOREM 18 *(Structure Theorem for RBSP) Let $\epsilon > 0$ be any constant. Let the minimum nonzero internode distance in a RBSP instance be 8 and let L_R be the size of its red bounding box. Let shifts $0 \leq a, b \leq L_R$ be chosen randomly. Then with probability at least 1/2, there is a separating polygon of cost at most $(1 + \epsilon)OPT$ that is (m, r)-light with respect to the dissection with shift (a, b), where $m = O(c \log L_R)$ and $r = O(1/\epsilon)$.*

STEP 3: DYNAMIC PROGRAMMING. Assuming the veracity of Theorem 18, we now describe the dyamic programming algorithm that finds the optimum (m, r)-light separating polygon. The algorithm is very similar to the algorithm for DRMST. The basic subproblems of the dynamic programming involve the following inputs: (a) A quadtree square, Q. (b) A set of $k \leq 4r$ portals, where the separating polygon crosses the boundary of Q. (c) A template, which consists of pairs of portals that are connected by a union of edges of the separating polygon lying in Q. (d) A 2-coloring, which indicates whether each region defined by lines segments in Q is "inside" the polygon (a red region) or "outside"

the polygon (a blue region.)

As in the dynamic programming algorithm for DRMST, the size of the lookup table is the number of subproblems. Note the number of possible colorings for input (d) is 2. The reason for this is that the colors of all regions inside Q are determined by specifying the color of any one region inside Q. Thus, the number of subproblems is $O(n \log n) \times (\sum_{k=1}^{r} \binom{m}{k}) \times (4r)^{4r} \times 2$, which is $O(n(\log n)^{1/\epsilon})$.

To solve each subproblem, we proceed in the usual bottom-up fashion. We first describe how to solve the base case, which involves quadtree squares that contain a single point (or multple points with internode distance zero.)

Fix a template and a color scheme. For the quadtree square with a single node, the template is valid only if the pairs of connected portals can be connected by straight line segments that do not intersect each other. Thus, in order to solve the base case for valid templates, we just connect the specified portals with straight line segments. If the single node in the quadtree square is contained in a region of the correct color, we are done. Otherwise, we may have to "bend" some of the added line segments in order to establish the correct coloring of the square. Note that if there are nodes of different color with internode distance 0, we simply make sure the separating polygon passes through these nodes. The base case computations can be performed by enumeration in constant time, since the number of pairs of portals in the template is bounded by the constant $(4r)^{4r}$.

For all other quadtree squares, we check the lookup table for the solutions to the subproblems in the four child squares. We then match these child solutions so that their templates and color schemes are consistent, and choose the lowest cost solution.

All that remains is the proof of Theorem 18.

4 Patching Lemma for RBSP

The proof of Theorem 18 follows exactly the proof of Theorem 2 found in Section 2.2 of [2]. (the proof is quite involved and we refer the reader to Arora's paper.) The only difference is that all references to Lemma 3 of [2], the Patching Lemma for TSP, are now replaced with references to the Patching Lemma for RBSP, which we prove in this section.

LEMMA 19 (PATCHING LEMMA FOR RBSP) *Let S be any line segment of length s and P be the boundary of a simple separating polygon that crosses S at least three times. Then we can break P at all but two of these places and add segments lying on S of total length at most $4s$, so that we have a separating polygon whose boundary P' crosses S in at most two places.*

PROOF: We assume without loss of generality that S is a horizontal grid line. We describe a five step patching algorithm that converts P into P' that satisfies the conditions of the lemma.

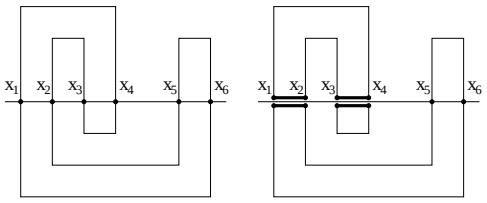

Figure 6: On the left, the original polygon. On the right, the polygon after Step 3.

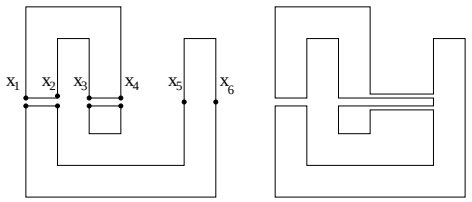

Figure 7: On the left, the polygon after Step 3. On the right, the polygon after Step 4.

STEP 1: Label the $t > 2$ crossings of S, from left to right, as the points x_1, x_2, ...,x_t.

We know that one side of each line of P lies on the "inside" and the other lies on the "outside" of the polygon. Since these inside and outside regions must alternate, we have one of two scenarios:

A. Segments $\overline{x_1x_2}$, $\overline{x_3x_4}$, $\overline{x_5x_6}$, ... are all contained in the interior of the polygon (and $\overline{x_2x_3}$, $\overline{x_4x_5}$, ... are not.)

B. Segments $\overline{x_2x_3}$, $\overline{x_4x_5}$, $\overline{x_6x_7}$, ... are all contained in the interior of the polygon (and $\overline{x_1x_2}$, $\overline{x_3x_4}$, ... are not.)

STEP 2: If we have Scenario A, break P at x_1, x_2, x_3, ..., x_{t-2}. If t is odd, break P at x_{t-1} as well. If we have Scenario B, break P at x_2, x_3, ..., x_{t-1}. If t is odd, break P at x_t as well.

STEP 3: For Scenario A, if t is odd, add line segments $\overline{x_1x_2}$, $\overline{x_3x_4}$, $\overline{x_5x_6}$, ..., $\overline{x_{t-2}x_{t-1}}$ to both sides of S (although we refer to both of these segments as lying on S, we think of them as having a small gap between them; one lies slightly above S, the other slightly below S.) If t is even, add line segments $\overline{x_1x_2}$, $\overline{x_3x_4}$, $\overline{x_5x_6}$, ..., $\overline{x_{t-3}x_{t-2}}$. For Scenario B, add line segments $\overline{x_2x_3}$, $\overline{x_4x_5}$, $\overline{x_6x_7}$, ... See Figure 6.

Step 3 has patched the open ends of the original simple polygon, so that we have a union of connected components with well-defined insides and outsides (this assertion will be proved below.) All points of one color are inside and all points of the other color are outside. The patching has not introduced any

crossings of the new boundary (i.e. the polygons) with itself, nor any new intersections of the new boundary with S.

STEP 4: **do** the following, first for connected components that intersect the region above S, then for all components that remain:

Let the connected components of interest be P_1, P_2, ..., P_k. Let y_i be the point x_j, such that j is the largest j such that P_i touches x_j. Let $y_i' = x_{j+1}$. Note for exactly one i, y_i' does not exist. Assume without loss of generality that this $i = k$. Then, add two copies of the edges $\overline{y_1 y_1'}$, $\overline{y_2 y_2'}$, ..., $\overline{y_{k-1} y_{k-1}'}$. See Figure 7.

CLAIM 1: *After Step 3, the graph has been modified into a union of disjoint, simple polygons.*
PROOF: The addition of line segments in Step 3 ensures that each of the x_i's has degree two. We denote the set of added line segments by E. Since these segments did not introduce any crossings into the graph, after Step 3 the graph can be decomposed into a family of disjoint, simple cycles. We denote this family by $\mathcal{C}$. Since these cycles are Jordan curves, they have well-defined interiors. We define the interior of a cycle c to contain the region inside the cycle, but not to include the cycle itself. We argue that a cycle in $\mathcal{C}$ cannot be contained in the interior of a different cycle in $\mathcal{C}$. This fact will imply that the graph has been decomposed into a set of connected components, each of which is a simple polygon. Note that these polygons contain all the red points, and none of the blue points.

The removal of the x_i's in Step 1 caused the original polygon to be broken up into a family of disjoint paths. Adding line segments in E connects the paths together to form the cycles in $\mathcal{C}$. It follows from this observation that every cycle in $\mathcal{C}$ must contain at least one of the edges added in E. Let $P \in \mathcal{C}$ a cycle in the graph. We want to show that P does not contain any segment of E in its interior. By hypothesis,the original separating polygon was simple, and thus the line segments on S between line segments in E were not contained on the boundary of the original separating polygon. After removing the crossings and adding the line segments in E, none of the cycles in $\mathcal{C}$ may contain segments of E in their interior.

The facts that no cycles in $\mathcal{C}$ intersect, but each cycle has at least one edge in E, implies that the interior of P does not contain any cycle of $\mathcal{C}$ (since the interior cannot contain an edge of E.) □

CLAIM 2: *After Step 4, all the simple separating polygons created in Step 3 have been "merged" into a single polygon that separates red from blue.*
PROOF: Let us first consider what happens when we add two copies of the edge $\overline{y_i y_i'}$, which is the edge between P_i and P_{i+1}. We in fact have merged P_i and P_{i+1} into a single polygon. The boundary of this polygon is: (1) starting from y_i, traverse the cycle defined by the boundary of P_i, returning to y_i. (2) traverse one copy of $\overline{y_i y_i'}$ to reach y_i'. (3) traverse the cycle defined by the boundary of P_{i+1}, returning to y_i'. (4) traverse the other copy of $\overline{y_i y_i'}$ to return to y_i.

Thus, we have merged P_i and P_{i+1} into a single polygon. Adding all edges in Step 4 will merge all polygons into a single polygon that separates red form blue. □

Note that the polygon created after Step 4 is not simple; it necessarily intersects itself, since edges added in Step 4 lie directly on top of each other. This problem is remedied by slightly perturbing the polygon by adding an aribitrarily small vertical separation between these pairs of edges, which will remove the polygon's self intersections, leaving us with a simple separating polygon.

STEP 5: Perturb the polygon in order to remove self-intersections.
□

5 Conclusions

Can we design approximation schemes for other geometric problems that involve complicated topology? Minimum-weight Steiner triangulation seems like the next obvious candidate. We only know of a 316-approximation due to Eppstein. For a survey of this and other problems with topological constraints, see Bern and Eppstein [7]. For all of these problems a first step would be to prove theorems about the structure of the optimum solutions (analogous to Lemma 4.)

We do not currently see a way to reduce the running time of our approximation scheme from quasipolynomial to polynomial.

References

[1] S. Arora Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proc. 37th IEEE Symp. on Foundations of Computer Science*, 1996.

[2] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *JACM* **45**(5) 753-782, 1998.

[3] S. Arora. Approximation schemes for NP-hard geometric optimization problems: A survey. *Math Programming*, 2003 Available from `www.cs.princeton.edu/~arora`.

[4] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. *Proc. 34th ACM Symposium on Theory of Computing*, 1999.

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

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

[7] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In [10].

[8] T. Chan Euclidean Bounded-Degree Spanning Tree Ratios *Proc. 19th ACM Symposium on Computational Geometry*, pp 11-19, 2003.

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

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

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

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

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

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

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

[16] C. H. Papadimitriou and U. V. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms* **5**(1984), pp. 231–246.

[17] B. Raghavachari. Algorithms for finding low degree structures. In [10]

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