Minimum-Cost Coverage of Point Sets by Disks*

Helmut Alt† Esther M. Arkin‡ Hervé Brönnimann§ Jeff Erickson¶ Sándor P. Fekete‖
Christian Knauer† Jonathan Lenchner** Joseph S. B. Mitchell‡ Kim Whittlesey††

Abstract

We consider a class of geometric facility location problems in which the goal is to determine a set X of disks given by their centers (t_j) and radii (r_j) that cover a given set of demand points $Y \subset \mathbb{R}^2$ at the smallest possible cost. We consider cost functions of the form $\sum_j f(r_j)$, where $f(r) = r^\alpha$ is the cost of transmission to radius r. Special cases arise for $\alpha = 1$ (sum of radii) and $\alpha = 2$ (total area); power consumption models in wireless network design often use an exponent $\alpha > 2$. Different scenarios arise according to possible restrictions on the transmission centers t_j, which may be constrained to belong to a given discrete set or to lie on a line, etc.

We obtain several new results, including (a) exact and approximation algorithms for selecting transmission points t_j on a given line in order to cover demand points $Y \subset \mathbb{R}^2$; (b) approximation algorithms (and an algebraic intractability result) for selecting an optimal line on which to place transmission points to cover Y; (c) a proof of NP-hardness for a discrete set of transmission points in $\mathbb{R}^2$ and any fixed $\alpha > 1$; and (d) a polynomial-time approximation scheme for the problem of computing a *minimum cost covering tour* (MCCT), in which the total cost is a linear combination of the transmission cost for the set of disks and the *length* of a tour/path that connects the centers of the disks.

ACM Classification: F.2.2 Nonnumerical Algorithms and Problems.

AMS Classification: 68Q25, 68U05, 90C27.

Keywords: Covering problems, tour problems, geometric optimization, complexity, approximation.

1 Introduction

The problem. We study a geometric optimization problem that arises in wireless network design, as well as in robotics and various facility location problems. The task is to select a number of locations t_j for the base station antennas (*servers*), and assign a transmission range r_j to each t_j, in order that each $p_i \in Y$ for a given set $Y = \{p_1, \dots, p_n\}$ of n demand points (*clients*) is covered. We say that client p_i is covered if and only if p_i is within range of some transmission point t_{j_i}, i.e., $d(t_{j_i}, p_i) \le r_{j_i}$. The resulting cost per server is some known function f, such as $f(r) = r^\alpha$. The goal is to minimize the total cost, $\sum_j f(r_j)$, over all placements of at most k servers that cover the set Y of clients. In the *discrete* version, a set X of m potential locations for the servers is specified.

In the context of modeling the energy required for wireless transmission, it is common to assume a superlinear ($\alpha > 1$) dependence of the cost on the radius; in fact, physically accurate simulation often requires superquadratic dependence ($\alpha > 2$). A quadratic dependence ($\alpha = 2$) models the total area of the served region, an objective arising in some applications. A linear dependence ($\alpha = 1$) is sometimes assumed, as in Lev-Tov and Peleg [18], who study the base station coverage problem, minimizing the sum of radii. The linear case is important to study not only in order to simplify the problem and gain insight into the general problem, but also to address those settings in which the linear cost model naturally arises [10,20]. For example,

*We thank all of the participants of the McGill-INRIA International Workshop on Limited Visibility, at the Bellairs Research Institute of McGill University, where this research was originated. We acknowledge valuable conversations with Sue Whitesides, Nancy Amato, Beppe Liotta, and other workshop participants and heartily thank the organizers Sue Whitesides and Hazel Everett for facilitating and enabling a wonderful working environment. E. Arkin is partially supported by grants from the National Science Foundation (CCR-0098172, CCF-0431030). H. Brönnimann and J. Lenchner are partially supported by a grant from the National Science Foundation (Career grant CCR-0133599). J. Mitchell is partially supported by grants from the National Science Foundation (CCR-0098172, ACI-0328930, CCF-0431030, CCF-0528209), the U.S.-Israel Binational Science Foundation (2000160), Metron Aviation, and NASA Ames (NAG2-1620).

†Institute of Computer Science, Freie Universität Berlin, D-14195 Berlin, Germany. {alt, knauer}@inf.fu-berlin.de.
‡Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794, USA. {estie, jsbm}@ams.sunysb.edu.
§Computer and Information Science, Polytechnic University, Brooklyn, NY 11201, USA. hbr@poly.edu.
¶Department of Computer Science, University of Illinois, Urbana, IL 61801, USA. jeffe@cs.uiuc.edu.
‖Department of Mathematical Optimization, Braunschweig University of Technology, D-38106 Braunschweig, Germany. s.fekete@tu-bs.de.
**IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA. lenchner@us.ibm.com.
††Department of Mathematics, University of Illinois, Urbana, IL 61801, USA. kwhittle@math.uiuc.edu.

the model may be appropriate for a system with a narrow-angle beam whose direction can either rotate continuously or adapt to the needs of the network. Another motivation for us comes from robotics, in which a robot is to map or scan an environment with a laser scanner [13, 14]: For a fixed spatial resolution of the desired map, the time it takes to scan a circle corresponds to the number of points on the perimeter, i.e., is proportional to the radius.

Our problem is a type of clustering problem, recently named *min-size k-clustering* by Bilò et al. [7]. Clustering problems tend to be NP-hard, so most efforts, including ours, are aimed at devising an approximation algorithm or a polynomial-time approximation scheme (PTAS).

We also introduce a new problem, which we call *minimum cost covering tour* (MCCT), in which we combine the problem of finding a short tour and placing covering disks centered along it. The objective is to minimize a linear combination of the tour length and the transmission/covering costs. The problem arises in the autonomous robot scanning problem [13, 14], where the covering cost is linear in the radii of the disks, and the overall objective is to minimize the total time of acquisition (a linear combination of distance travelled and sum of scan radii). Another motivation is the distribution of a valuable or sensitive resource: There is a trade-off between the cost of broadcasting from a central location (thus wasting transmission or risking interception) and the cost of travelling to broadcast more locally, thereby reducing broadcast costs but incurring travel costs.

Location Constraints. In the absence of constraints on the server locations, it may be optimal to place one server at each demand point. Thus, we generally set an upper bound, k, on the number of servers, or we restrict the possible locations of the servers. Here, we consider two cases of location constraints:

(1) Servers are restricted to lie in a discrete set $\{t_1, \dots, t_m\}$; or

(2) Servers are constrained to lie on a line (which may be fully specified, or may be selected by the optimization).

Our results. We provide a number of new results, some improving previous work, some giving the first results of their kind.

In the discrete case studied by Lev-Tov and Peleg [18], and Biló et al. [7], we give improved results. For the discrete 1D problem where $Y \subseteq \mathbb{R}$, we improve their 4-approximation to a linear-time 3-approximation by using a "Closest Center with Growth" (CCG) algorithm, and, as an alternative to the previous $O((n+m)^3)$ algorithm [18], we give a near-linear-time 2-approximation that uses a "Greedy Growth" (GG) algorithm.

In the general 2D case with clients $Y \subset \mathbb{R}^2$, we strengthen the hardness result of Biló et al. [7] by showing that the discrete problem is already hard for any superlinear cost function, i.e., $f(r) = r^\alpha$ with $\alpha > 1$. Furthermore, we generalize the min-size clustering problem in two new directions. On the one hand, we consider less restrictive server placement policies. For instance, if we only restrict the servers to lie on a given fixed line, we give a dynamic programming algorithm that solves the problem exactly, in time $O(n^2 \log n)$ for any L_p metric in the linear cost case, and in time $O(n^4 \log n)$ in the case of superlinear non-decreasing cost functions. For simple approximations, our algorithm "Square Greedy" (SG) gives in time $O(n \log n)$ a 3-approximation to the square covering problem with any linear or superlinear cost function. A small variation, "Square Greedy with Growth" (SGG), gives a 2-approximation for a linear cost function, also in time $O(n \log n)$. The results are also valid for covering by L_p disks for any p, but with correspondingly coarser approximation factors.

If the servers are restricted to lie on a horizontal line, but the location of this line may be chosen freely, then we show that the exact optimal position (with $\alpha = 1$) is not computable by radicals, using an approach similar to that of Bajaj [5, 6] in addressing the unsolvability of the Fermat-Weber problem. On the positive side, we give a fully polynomial-time approximation scheme (FPTAS) requiring time $O((n^3/\varepsilon)\log n)$ if $\alpha = 1$ and time $O((n^4/\varepsilon)\log n)$ if $\alpha > 1$.

For servers on an unrestricted line, of any slope, and $\alpha = 1$, we give $O(1)$-approximations (4-approximation in $O(n^4 \log n)$ time, or $8\sqrt{2}$-approximation in $O(n^3 \log n)$ time) and an FPTAS requiring time $O((n^5/\varepsilon^2)\log n)$.

We give the first algorithmic results for the new problem, minimum cost covering tour (MCCT), which we introduce. Given a set $Y \subseteq \mathbb{R}^2$ of n clients, our goal is to determine a polygonal tour T and a set X of k disks of radii r_j centered on T that cover Y while minimizing the cost length$(T) + C\sum r_i^\alpha$. Our results are for $\alpha = 1$. The ratio C represents the relative cost of touring versus transmitting. We show that MCCT is NP-hard if C is part of the input. At one extreme, if C is small then the optimum solution is a single server placed at the circumcenter of Y (we can show this to be the case for $C \le 4$). At the other extreme (if C very large), the optimum solution is a TSP among the clients. For any fixed value of $C > 4$, we present a PTAS for MCCT, based on a novel extension of the m-guillotine methods of [19].

Related work. There is a vast family of clustering problems, among which are the *k-center* problem in which one minimizes $\max_j r_j$, the *k-median* problem in which one minimizes $\sum_i d(p_i, t_{j_i})$, and the *k-clustering* problem in which one minimizes the maximum over all clusters of the sum of pairwise distances between points in that cluster. For the geometric instances of these related clustering problems, refer to the survey by Agarwal and Sharir [1]. When k is fixed, the optimal solution can be found in time $O(n^k)$ using brute force. In the plane, one of the only results for the min-size clustering problem is a small improvement for $k = 2$ by Hershberger [16], in subquadratic time $O(n^2/\log\log n)$. Approximation algorithms and schemes have been proposed, particularly for geometric instances of these problems (e.g., [4]). Clustering for minimizing the sum of radii was studied for points in metric spaces by Charikar and Panigrahy [9], who present an $O(1)$-approximation algorithm using at most k clusters.

For the linear-cost model ($\alpha = 1$), our problem has been considered recently by Lev-Tov and Peleg [18] who give an $O((n+m)^3)$ algorithm when the clients and servers all lie on a given line (the 1D problem), and a linear-time 4-approximation in that case. They also give a PTAS for the two-dimensional case when the clients and servers can lie anywhere in the plane. Bilò et al. [7] show that the problem is NP-hard in the plane for the case $f(r) = r^\alpha$, $\alpha \geq 2$, either when the sets X and Y are given and k is left unspecified ($k = n$), or when k is fixed but then $X = Y$. They give a PTAS for the linear cost case ($\alpha = 1$) and a slightly more involved PTAS for a more general problem in which the cost function is superlinear, there are fixed additive costs associated with each transmission server and there is a bound k on the number of servers.

There are many problems dealing with covering a set of clients by disks of *given* radius. Hochbaum and Maass [17] give a PTAS for covering with a minimum number of disks of fixed radius, where the disk centers can be taken anywhere in the plane. They introduce a "grid-shifting technique," which is used and extended by Erlebach et al. [12]. Lev-Tov and Peleg [18] and Bilò et al. [7] extend the method further in obtaining their PTAS results for the discrete version of our problem.

When a discrete set X of potential server locations is given, Gonzalez [15] addresses the problem of maximizing the number of covered clients while minimizing the number of servers supplying them, and he gives a PTAS for such problems with constraints such as bounded distance between any two chosen servers. In [8], a polynomial-time constant approximation is obtained for choosing a subset of minimum size that covers a set of points among a set of candidate disks (the radii can be different but the candidate disks must be given).

The closest work to our combined tour/transmission cost (MCCT) is the work on covering tours: the "lawn mower" problem [2], and the TSP with neighborhoods [3, 11], each of which has been shown to be NP-hard and has been solved with various approximation algorithms. In contrast to the MCCT we study, the radius of the "mower" or the radius of the neighborhoods to be visited is specified in advance.

2 Scenario (1): Server Locations Restricted to a Discrete Set

2.1 The one-dimensional discrete problem with linear cost

Consider the case of m fixed server locations $X = \{t_1, ..., t_m\}$, n client locations $Y = \{p_1, ..., p_n\}$, and a linear ($\alpha = 1$) cost function, with clients and servers all located along a fixed line. Without loss of generality, we may assume that X and Y are sorted in the same direction, at an extra cost of $O((n+m)\log(n+m))$. Lev-Tov and Peleg [18] give an $O((n+m)^3)$ dynamic programming algorithm for finding an exact solution. Bilò et al. [7] show that the problem is solvable in polynomial time for any value of α by reducing it to an integer linear program with a totally unimodular constraint matrix. The complexities of these algorithms, while polynomial, is high. Lev-Tov and Peleg also give a simple "closest center" algorithm (CC) that gives a linear-time 4-approximation. We improve to a 3-approximation in linear time, and a 2-approximation in $O(m + n\log m)$ time.

Closest Center with Growth (CCG) Algorithm: Process the clients $\{p_1, ..., p_n\}$ from left to right keeping track of the rightmost extending disk. Let ω_R denote the rightmost point of the rightmost extending disk, and let R denote the radius of this disk. (In fact the rightmost extending disk will always be the last disk placed.) If ω_R is equal to, or to the right of the next client processed, p_i, then p_i is already covered so ignore it and proceed to the next client. If p_i is not yet covered, consider the distance of p_i to ω_R compared with the distance of p_i to its closest center $\hat{t}_i$. If the distance of p_i to ω_R is less than or equal to the distance of p_i to its closest center $\hat{t}_i$, then grow the rightmost extending disk just enough to capture p_i. Otherwise use the disk centered at $\hat{t}_i$ of radius $|p_i - \hat{t}_i|$ to cover p_i.

Lemma 1 *CCG yields a* 3*-approximation to OPT in* $O(n+m)$ *time.*

Proof. Consider any disk D in OPT. We attribute to each client a segment J_i as follows. If, in the execution of CCG, the client p_i was not used because it had already been covered, we set $J_i = \emptyset$. If p_i was captured by placing a disk centered at the closest center, $\hat{t}_i$ to p_i then set $J_i = \{[\hat{t}_i, p_i] \text{ if } \hat{t}_i \leq p_i,\ [p_i, \hat{t}_i] \text{ if } p_i < \hat{t}_i\}$. On the other hand, if p_i was captured by growing an existing disk with initial rightmost point ω_R, let J_i denote the half-open interval through which this rightmost point moved out, i.e. $J_i = (\omega_R, p_i]$. Observe that $J_i \cap J_j = \emptyset$ as long as $i \neq j$ and that the sum of the lengths of the J_i equals the sum of the radii of disks in the CCG cover. The leftmost and rightmost J_i cannot extend more than radius(D) to the left of D or radius(D) to the right of D.

Let t_D denote the center of D. At most one J_i corresponding to a client in D extends outward to the right from the right edge of D. If there is no such right-most interval, the we clearly have at most a 3-approximation. Thus assume there is such an interval, and call it J_R. J_R corresponds to a center, not growth, since it emanates from the right. Call the associated client p_R. If there is a client p_i to the right of t_D not contained in J_R then length$(J_R) <$ radius$(D) - d(t_D, p_i)$ since otherwise in the algorithm we would have grown the disk containing p_i to capture p_R, rather than allow it to be captured by a center. It follows that the coverage by disks in CCG to the right of t_D has sum of radii at most radius(D). The 3-approximation follows. □

If we consider a single disk D with clients p_L and p_R on the left and right edges of D, associated centers $\hat{x}_L$, $\hat{x}_R$ at distances respectively radius(D)$-\varepsilon$ to the left and radius(D)$-\varepsilon$ to the right, along with a dense set of clients in the left hand half of D we see that 3 is the best possible constant for CCG.

Greedy Growth (GG) Algorithm: Start with a disk with center at each server all of radius zero. Now, amongst all clients, find the one which requires the least radial disk growth to capture it. Repeat until all clients are covered. An efficient implementation uses a priority queue to determine the client that should be captured next. One can set up the priority queue in $O(m)$ time. Note that the priority queue will never have more than $2m$ elements, and that each p_i eventually gets captured, either from the right or from the left. Each capture can be done in time $O(\log m)$ for a total running time of $O(m + n\log m)$.

Lemma 2 *GG yields a* 2*-approximation to OPT in* $O(m+n\log m)$ *time.*

Proof. Define intervals J_i as follows: when capturing a client p_i from a server t_j whose current radius (prior to capture) is r_j, let $J_i = (t_j + r_j, p_i]$ if $p_i > t_j$, and $J_i = [p_i, t_j - r_j)$ otherwise. Our first trivial yet crucial observation is that $J_i \cap J_k = \emptyset$ if $i \neq k$. Also note that the sum of the lengths of the J_i is equal to the sum of the radii in the GG cover.

Consider now a fixed disk D in OPT, centered at t_D, and the list of intervals J_i whose p_i is inside D. As before, at most one such J_i extends outward to the right from the right edge of D. If so, call it J_R, and define J_L symmetrically. If J_R exists, it cannot extend more than radius(D) to the right of D. Let $\lambda = \text{length}(J_R)$. We argue that there is an interval of length λ in D, to the right of t_D, which is free of J_i's. It follows that there is at most radius(D) worth of segments to the right of t_D. Of course, this is also true if J_R does not exist. By symmetry, there is also radius(D) worth of segments to the left of t_D, whether J_L exists or not, yielding the claimed 2-approximation.

Assume J_R exists. Then the algorithm successively extends J_R by growth to the left up to some maximum point (possibly stopping right at p_R). Since the growth could have been induced by clients to the right of J_R, that maximum point is not necessarily a client. There is, however, some client inside D that is captured last in this process. This client p_i (possibly p_R) cannot be within λ of t_D, since otherwise it would have been captured prior to the construction of J_R.

If there is no client between t_D and p_i we are done, since then there could be no interval J_k in between. Thus consider the client p_{i-1} just to the left of p_i. Suppose $d(p_{i-1}, p_i) \geq \lambda$. Then, if p_{i-1} is eventually captured from the left, we would have the region between p_{i-1} and p_i free of J_k's and be done. On the other hand, if p_{i-1} is captured from the right, it must be captured by a server between p_{i-1} and p_i, and that server is at least λ to the left of p_i since otherwise p_i would be captured by that server prior to p_R. This leaves the distance from the server to p_i free of J_k's.

Hence the only case of concern is if $d(p_{i-1}, p_i) < \lambda$. Clearly p_{i-1} must not have been captured at the time when p_R is captured since otherwise p_i would have been captured before p_R, contradicting the assumption that p_i is captured by growth leftward from p_R. Similarly, there cannot be a server between p_{i-1} and p_i, since otherwise both p_{i-1} and p_i would be captured before p_R. Together with the definition of p_i, this implies that p_{i-1} is captured from the left. Therefore, to the left of p_{i-1}, there must be one or more intervals $\{J_{l_i}\}$ whose length is at least λ that are constructed before p_{i-1} is captured. Similarly, to the right of p_i, there must be some one or more intervals $\{J_{r_j}\}$ whose length is at least λ, constructed before p_i is captured. However, either the last J_{l_i} is placed before the last J_{r_j} or vice versa. In the first case, there are no λ length obstructions left in the left-hand subproblem, so p_{i-1} will be covered, and with λ length obstructions remaining in the right subproblem, p_i will be captured by growth rightward. The second case is symmetrical to the first. In either case we have a contradiction. □

To see that the factor 2 is tight, just consider servers at $-2+\varepsilon, 0$ and $2-\varepsilon$ and clients at -1 and 1.

2.2 Hardness of the two-dimensional discrete problem with superlinear cost

In 2D, we sketch an NP-hardness proof, for any $\alpha > 1$. This strengthens the NP-hardness proof of [7], which only works in the case $\alpha \geq 2$. Our proof is based on PLANAR 3SAT: Only use a subset of the set of critical locations as candidate locations by only choosing the points that are "halfway" between two adjacent client points along a variable gadget. This allows only two perfect matchings on each variable gadget as locally optimal solutions; these matchings map to truth assignments in a canonical way. A satisfying truth assignment on a variable allows picking up an additional point at a clause gadget, yielding an inexpensive solution. See the Appendix for an illustration of clause gadgets.

Theorem 3 *For any a fixed* $\alpha > 1$*, let the cost function of a circle of radius r be* $f(r) = r^{\alpha}$*. Then it is NP-hard to decide whether a discrete set of n clients in the plane, and a discrete set of m potential transmission points allow a cheap set of circles that covers all demand points.*

3 Scenario (2): Server Locations Restricted to a Line

3.1 Servers along a fixed horizontal line

3.1.1 Exact solutions Suppose that the servers are required to lie on a fixed horizontal line, which we take without loss of generality to be the x-axis. Such a restriction could arise naturally (e.g., the servers must be connected to a power line, must lie on a highway, or in the main corridor in a building). In addition, this case must be solved first before attempting to solve the more general problem—along a polygonal curve.

In this section, we describe dynamic programming algorithms to compute a set of server points of minimum total cost. For notational convenience, we assume that the clients Y are indexed in left-to-right order. Without loss of generality, we also assume that all the clients lie on or above the x-axis, and that no two clients have the same x-coordinate. (If a client p_i lies directly above another client p_j, then any circle enclosing p_i also encloses p_j, so we can remove p_j from Y without changing the optimal cover.)

Let us call a circle C *pinned* if it is the leftmost smallest axis-centered circle enclosing some fixed subset of clients. Equivalently, a circle is pinned if it is the leftmost smallest circle passing through a chosen client or a chosen pair of clients. Under any L_p metric, there are at most $O(n^2)$ pinned circles. As long as the cost function f is non-decreasing, there is a minimum-cost cover consisting entirely of pinned circles.

Linear Cost. If the cost function f is linear (or sublinear), we easily observe that the circles in any optimum solution must have disjoint interiors. (If two axis-centered circles of radius r_i and r_j intersect, they lie in a larger axis-centered circle of radius at most $r_i + r_j$.) In this case, we can give a straightforward dynamic programming algorithm that computes the optimum solution under any L_p metric.

The algorithm given in Figure 1 (left) finds the minimum-cost cover by disjoint pinned circles, where distance is measured using any L_p metric. We call the rightmost point enclosed by any pinned circle C the *owner* of C.

If we use brute force to compute the extreme points enclosed by each pinned circle and to test whether any points lie directly above a pinned circle, this algorithm runs in $O(n^3)$ time. With some more work, however, we can improve the running time by nearly a linear factor.

This improvement is easiest in the L_∞ metric, in which circles are axis-aligned squares. Each point p_i is the owner of exactly i pinned squares: the unique axis-centered square with p_i in the upper right corner, and for each point p_j to the left of p_i, the leftmost smallest axis-centered square with p_i and p_j on its boundary. We can easily compute all these squares, as well as the leftmost point enclosed by each one, in $O(i \log i)$ time. (To simplify the algorithm, we can actually ignore any pinned square whose owner does not lie on its right edge.) If we preprocess P into a priority search tree in $O(n \log n)$ time, we can test in $O(\log n)$ time whether any client lies directly above a horizontal line. The overall running time is now $O(n^2 \log n)$.

For any other L_p metric, we can compute the extreme points enclosed by all $O(n^2)$ pinned circles in $O(n^2)$ time using the following duality transformation. If C is a circle centered at $(x,0)$ with radius r, let C^* be the point (x,r). For each client p_i, let $p_i^* = \{C^* \mid p \in C\}$, and let $Y^* = \{p_i^* \mid p_i \in Y\}$. We easily verify that each set p_i^* is an infinite x-monotone curve. (Specifically, in the Euclidean metric, the dual curves are hyperbolas with asymptotes of slope ± 1.) Moreover, any two dual curves p_i^* and p_j^* intersect exactly once; i.e., Y^* is a set of pseudo-lines. Thus, we can compute the arrangement of Y^* in $O(n^2)$ time. For each pinned circle C, the dual point C^* is either one of the clients p_i or a vertex of the arrangement of dual curves Y^*. A circle C encloses a client p_i if and only if the dual point C^* lies on or above the dual curve p_i^*. After we compute the dual arrangement, it is straightforward to compute the leftmost and rightmost dual curves below every vertex in $O(n^2)$ time by depth-first search.

Finally, to test efficiently whether any points lie directly above an axis-centered (L_p) circle, we can use the following two-level data structure. The first level is a binary search tree over the x-coordinates of Y. Each internal node v in this tree corresponds to a canonical vertical slab S_v containing a subset p_v of the clients. For each node v, we partition the x-axis into intervals by intersecting it with the furthest-point Voronoi diagram of p_v, in $O(|p_v| \log |p_v|)$ time. To test whether any points lie above a circle, we first find a set of $O(\log n)$ disjoint canonical slabs that exactly cover the circle, and then for each slab S_v in this set, we find the furthest neighbor in p_v of the center of the circle by binary search. The region above the circle is empty if and only if all $O(\log n)$ furthest neighbors are inside the circle. Finally, we can reduce the overall cost of the query from $O(\log^2 n)$ to $O(\log n)$ using fractional cascading. The total preprocessing time is $O(n \log^2 n)$.

Theorem 4 *Given n clients in the plane, we can compute in $O(n^2 \log n)$ time a covering by circles (in any fixed L_p metric) centered on the x-axis, such that the sum of the radii is minimized.*

Superlinear Cost. A similar dynamic programming algorithm computes the optimal covering under any superlinear (in fact, any *non-decreasing*) cost function f. As in the previous section, our algorithm works for any L_p metric. For the moment, we will assume that p is finite.

Although two circles in the optimal cover need not be disjoint, they cannot overlap too much. Clearly, no two circles in the optimal cover are nested, since the smaller circle would be redundant. Moreover, the highest point (or *apex*) of any circle in the optimal cover must lie outside all the other circles. If one circle A contains the apex of a smaller circle B, then the lune $B \setminus A$ is completely contained in an even smaller circle C whose apex is the highest point in the lune; it follows that A and B cannot both be in the optimal cover. See Figure 2(a).

To compute the optimal cover of Y, it suffices to consider subproblems of the following form. For each pinned circle C,

MinSumOfRadiusCircleCover(Y) :	MinSuperlinearCostCircleCover(Y, f) :
for every pinned circle C	sort the pinned circles from left to right by their centers
find the leftmost and rightmost points enclosed by C	$Cost[0] \leftarrow 0$
$Cost[0] \leftarrow 0$	for $j \leftarrow 1$ to $p+1$
for $i \leftarrow 1$ to n	$Cost[j] \leftarrow \infty$
$Cost[i] \leftarrow \infty$	for $i \leftarrow 1$ to $j-1$
for each pinned circle C owned by p_i	if C_i and C_j exclude each other's apices
if no points in P lie directly above C	and $B(C_i, C_j)$ is empty
$p_j \leftarrow$ leftmost point enclosed by C	$Cost[j] \leftarrow \min\{Cost[j], Cost[i] + f(radius(C_i))\}$
$Cost[i] \leftarrow \min\{Cost[i], Cost[j-1] + radius(C)\}$	return $Cost[p+1]$
return $Cost[n]$	

Figure 1. The dynamic programming algorithm: Left: linear cost; Right: superlinear cost function.

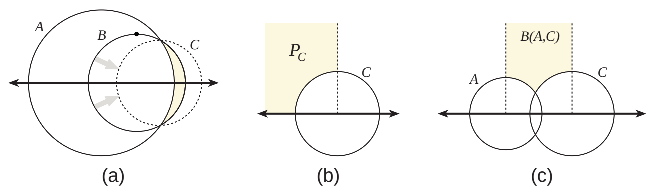

Figure 2. (a) The apex of each circle in the optimal cover lies outside the other circles. (b) The points Y_C lie in the shaded region. (c) If A and C are adjacent circles in the optimal covering, the shaded region $B(A,C)$ is empty.

let Y_C denote the set of clients outside C and to the left of its center; see Figure 2(b). Then for each pinned circle C, we have $cost(Y_C) = \min_A(f(radius(A)) + cost(Y_A))$, where the minimum is taken over all pinned circles A satisfying the following conditions: (1) The center of A is left of the center of C; (2) the apex of A is outside C; (3) the apex of C is outside A; and (4) A encloses every point in $Y_C \setminus Y_A$. The last condition is equivalent to there being no clients inside the region $B(A,C)$ bounded by the x-axis, the circles A and C, and vertical lines through the apices of A and C; see Figure 2(c).

Our dynamic programming algorithm (Figure 1 (right)) considers the pinned circles $C_1, C_2, \ldots, C_p$ in left to right order by their centers; that is, the center of C_i is left of the center of C_j whenever $i < j$. To simplify notation, let $Y_i = Y_{C_i}$. For convenience, we add two circles C_0 and C_{p+1} of radius zero, centered far to the left and right of Y, respectively, so that $Y_0 = \varnothing$ and $Y_{p+1} = Y$.

Implementing everything using brute force, we obtain a running time of $O(n^5)$. However, we can improve the running time to $O(n^4 \log n)$ using the two-level data structure described in the previous section, together with a priority search tree. The region $B(C_i, C_j)$ can be partitioned into two or three three-sided regions, each bounded by two vertical lines and either a circular arc or the x-axis. We can test each three-sided region for emptiness in $O(\log n)$ time.

Theorem 5 *Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a fixed non-decreasing cost function. Given n clients in the plane, we can compute in $O(n^4 \log n)$ time a covering by circles (in any fixed L_p metric) centered on the x-axis, such that the sum of the costs of the circles is minimized.*

The algorithm is essentially unchanged in the L_∞ metric, except now we define the apex of a square to be its upper right corner. It is easy to show that there is an optimal square cover in which no square contains the apex of any other square. Equivalently, we can assume without loss of generality that if two squares in the optimal cover overlap, the larger square is on the left. To compute the optimal cover, it suffices to consider subsets Y_C of points either directly above or to the right of each pinned square C. For any two squares A and C, the region $B(A,C)$ is now either a three-sided rectangle or the union of two three-sided rectangles, so we can use a simple priority search tree instead of our two-level data structure to test whether $B(A,C)$ is empty in $O(\log n)$ time.

However, one further observation does improve the running time by a linear factor: Without loss of generality, the rightmost box in the optimal cover of Y_C has the rightmost point of Y_C on its right edge. Thus, there are at most n candidate boxes C_i to test in the inner loop; we can easily enumerate these candidates in $O(n)$ time.

Theorem 6 *Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a fixed non-decreasing cost function. Given n clients in the plane, we can compute in $O(n^3 \log n)$ time a covering by axis-aligned squares centered on the x-axis, such that the sum of the costs of the squares is minimized.*

3.1.2 Fast and simple solutions In this section we describe simple and inexpensive algorithms that achieve constant factor approximations for finding a minimum-cost cover with disks centered along a fixed horizontal line L, using any L_p metric. The main idea for the proofs of this section is to associate with a given disk D in OPT, a set of disks in the approximate solution and argue that the set of associated disks cannot be more than a given constant factor cover of D, in terms of cumulative edge length, cumulative area, and so forth.

As in section 3.1.1, the case of L_∞ metric is the easiest to handle. By equivalence of all the L_p metrics, constant-factor c-approximations for squares will extend to constant-factor c'-approximations for L_p disks.

Square Greedy Cover Algorithm (SG): Process the client points in order of decreasing distance from the line L. Find the farthest point p_1 from L; cover p_1 with a square S_1 exactly of the same height as p_1 centered at the projection of p_1 on L. Remove all points covered by S_1 from further consideration and recurse, finding the next farthest point from L and so forth. In the case where two points are precisely the same distance from L, break ties arbitrarily.

Obviously, SG computes a valid covering of Y by construction.We begin the analysis with a simple observation.

Lemma 7 *In the SG covering, any point in the plane (not necessarily a client) cannot covered by more than two boxes.*

Proof. Suppose S_i and S_j are two squares placed during the running of SG and that $i < j$ so that S_i was placed before S_j. Then S_i cannot contain the center point of S_j since then S_j would not have had the opportunity to be placed, and similarly S_j cannot contain the center point of S_i. Now consider a point $p \in S_i \cap S_j$. If p were covered by a third square S_k then either one of $\{S_i, S_j\}$ would contain the center of S_k, or S_k would contain the center of one of $\{S_i, S_j\}$, neither of which is possible. $\square$

Theorem 8 *Given a set Y of n clients in the plane and any $\alpha \geq 1$, SG computes in time $O(n \log n)$ a covering of Y by axis-aligned squares centered on the x-axis whose cost is at most three times the optimal.*

Proof. Let $Y = \{p_1, \ldots, p_n\}$ and consider a square S in OPT. We consider those squares $\{S_{i_j}\}$ selected by SG corresponding to points $\{p_{i_j} : p_{i_j} \in S\}$, see Figure 3.1.2, and argue that these squares cannot have more than three times the total edge length

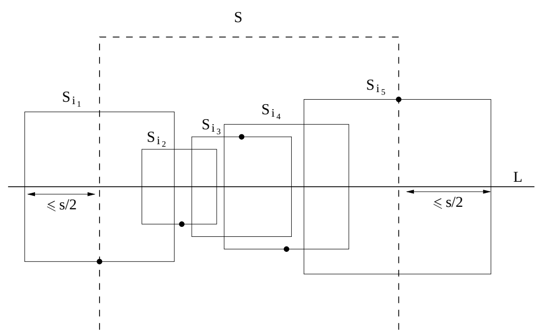

Figure 3. Squares of the SG algorithm inside a square of the optimal solution.

of S. The same will then follow for all of SG and all of OPT. The argument, without modification, covers the case of cost measured in terms of the sum of edge length raised to an arbitrary positive exponent $\alpha \geq 1$.

Arguing as in Lemma 7 it is easy to see that at most two boxes S_{i_j} associated with points $p_{i_j} \in S$ processed by SG actually protrude outside of S, one on the left and one on the right. Denote by r the total horizontal length of these protruding parts of squares, then $r \leq s$, the side length of S, since the side length of each protruding square is at most s and at most half of each square is protruding.

Because of Lemma 7 the total horizontal length of all nonprotruding parts of the squares S_{i_j} is at most $2s$, consequently all points covered by S in OPT are covered by a set of squares S_{i_j} in SG whose total (horizontal) edge length $\sum_j s_{i_j}$ is at most $3s$.

For exponents $\alpha > 1$ observe that $\sum_j s_{i_j} \leq 3s$ and $0 \leq s_{i_j} \leq s$ for all j implies that $\sum_j {s_{i_j}}^{\alpha} \leq 3s^{\alpha}$.

To analyze the running time of the algorithm we need some more details about the data structures used: Initially, sort the points by x-coordinate and separately by distance from the line L in time $O(n \log n)$ and process the points in order of decreasing distance from L. As the point p_i at distance d_i from L is processed, we throw away points which are within horizontal distance d_i from p_i. This takes time $O(\log n + k_i)$ time where k_i is the number of points within d_i from p_i. Since we do this up to n times with $k_1 + \cdots + k_l = n$ the total running time is $O(n \log n)$. $\square$

For the linear cost function, it is easy to modify the SG algorithm to get a 2-approximation algorithm.

Square Greedy with Growth Algorithm (SGG): Process the points as in SG. However, if capturing a point p_i by a square S_i would result in an overlap with already existing square S_j then, rather than placing S_i, grow S_j just enough to capture p_i, keeping the vertical edge furthest from p_i at the same point on L. If placing S_i would overlap two squares, grow the one which requires the smallest edge extension. Break ties arbitrarily.

A proof somewhat similar to that of Lemma 2 (given in the Appendix) shows that:

Theorem 9 *Given n clients in the plane, SGG computes in time $O(n \log n)$ a covering by axis-aligned squares centered on the x-axis whose cumulative edge length is at most twice the optimal.*

Unlike SG, SGG is not a constant factor approximation for area. Consider n consecutive points at height 1 separated one from the next by distance of $1+\varepsilon$. Processing the points left to right using SGG covers all points with one square of edge length $n + (n-1)\varepsilon$, and so area $O(n^2)$, while covering all points with n overlapping squares each of edge length 2, uses total area $4n$.

Finally, extending these results from squares to disks in any L_p metric is not difficult. Enclosing each square in the algorithm by an L_p disk leads to an approximation factor $3c^2$ for GG and $2c^2$ for SGG, where $c = p^{\alpha/p}$. In particular, for L_2 disks, this yields a $2\sqrt{2}$-approximation for $\alpha = 1$ and a 4-approximation for $\alpha = 2$.

3.2 Finding the best axis-parallel line

When the horizontal line ℓ is not given but its orientation is fixed, we first prove that finding the best line, even for $\alpha = 1$, is uncomputable, then in this linear case give a simple approximation, and finally a PTAS.

3.2.1 A hardness result – uncomputability by radicals Our approach is similar to the approach used by Bajaj on the unsolvability of the Fermat-Weber problem and other geometric optimization problems [5, 6].

Theorem 10 *Let $c(t) = \sum_i r_i$ denote the minimum cost of a cover whose centers lie on the line of equation $y = t$. There exists a set Y of clients such that, if t_0 is the value that minimizes $c(t)$, then t_0 is uncomputable by radicals.*

The proof proceeds by exhibiting such a point set and showing by differentiating $c(t)$ that t_0 is the root of a polynomial which is proven not to be solvable by radicals. The full proof is deferred to the Appendix.

3.2.2 Fast and simple constant-factor approximations The simple constant factor approximations for a fixed line can be extended to the case of approximations to the optimal solution on an arbitrary axis-parallel line with the same constant factors, though with an additional multiplicative factor of $O(n^2)$ added to the cost.

3.2.3 A PTAS for finding the best horizontal line Let d denote the distance between the highest and lowest point. Clearly, $d/2 \leq \text{OPT} \leq nd$. Cover this strip with n/ε regularly-spaced horizontal lines, where the space between successive lines is $\delta = d\varepsilon/2n$. For each line, run the exact dynamic programming algorithm. Consider the line that contains OPT. Moving this line by at most δ and enlarging OPT so that it still covers the points and remains centered on the line (an operation called *shifting*) increases the cost of a circle by at most δ, for a total of at most $\delta n = \varepsilon d/2 \leq \varepsilon$ OPT. Thus the algorithm computes a $(1+\varepsilon)$-approximation in running time $O((n^3/\varepsilon)\log n)$.

In order to generalize this result to the case $\alpha > 1$, let us write PSEUDO-OPT for the lowest cost of a solution on any of the regularly spaced horizontal lines, SHIFT for the result of shifting OPT to the closest of these lines, and $r_1, \ldots, r_m$ for the radii of the optimal set of disks. For an arbitrary power $\alpha \geq 1$, we have

$$\text{PSEUDO-OPT} \leq \text{SHIFT} \leq \sum_{i=1}^{m}(r_i+\delta)^\alpha \leq \sum_{i=1}^{m} r_i^\alpha + \delta\alpha\sum_{i=1}^{m}(r_i+\delta)^{\alpha-1} \leq \text{OPT}(1+\delta\alpha n 2^{2\alpha-1} n/d).$$

The latter step uses $\delta \leq d, r_i \leq d$ and $\text{OPT} \geq (d/2)^\alpha$. Choosing $\delta = \varepsilon d/(\alpha 2^{2\alpha-1} n)$ gives the desired $(1+\varepsilon)$-approximation.

Together with the results from previous sections we have:

Theorem 11 *Given n clients in the plane and a fixed $\alpha \geq 1$, there exists a PTAS for finding an optimally positioned horizontal line and a minimum-cost covering by disks centered on that line. It runs in time $O(n^3 \log n)$ in the linear case $(\alpha = 1)$ and $O(n^4 \log n)$ for $\alpha > 1$.*

3.3 Approximating the best line - any orientation

Finally, we sketch approximation results for selecting the best line whose orientation is not given. We give both a constant factor approximation and a PTAS for the linear cost case ($\alpha = 1$).

3.3.1 Fast and simple constant-factor approximations Given a line ℓ, we say that a set $\mathcal{D}$ of disks $D_1, \ldots, D_k$ is *ℓ-centered* if the centers of every disk C_j in $\mathcal{D}$ belongs to ℓ. Recall that the cost of $\mathcal{D}$ is the sum of all its radii.

Lemma 12 *Given $k \geq 1$, a line ℓ, an ℓ-centered set $\mathcal{D}$ of k disks that cover Y, and any point p_0 on ℓ, there exist $p' \in Y$ and an ℓ'-centered set $\mathcal{D}'$ of k disks that cover Y, where ℓ' is the line that joins p_0 and p', such that the cost of $\mathcal{D}'$ is at most twice that of $\mathcal{D}$.*

Proof. We will assume without loss of generality that ℓ is the x-axis, p_0 is the origin and that no other point in Y lies on the y-axis. The latter restriction can easily be enforced by a small perturbation. Let the coordinates of p_i be x_i and y_i, and let m_i denote the slope y_i/x_i of the line ℓ_i for $1 \leq i \leq n$. First, we reorder Y so that $|m_1| \leq \cdots \leq |m_n|$. In what follows we assume that $x_1 \geq 0$ and $y_1 0$. The other cases can be treated analogously.

For each disk $D_j = D(t_j, r_j)$ in $\mathcal{D}$, we construct a disk D'_j whose radius is $r'_j = 2r_j$ and center t'_j is obtained from t_j by rotating it around the origin counterclockwise by an angle $\tan^{-1}(m_1)$. The set $\mathcal{D}'$ of k disks thus defined is ℓ'-centered, where $\ell' = \{(x,y) \in \mathbb{R}^2 \mid y = m_1 x\}$ and $p_1 \in \ell'$. To see that $\mathcal{D}'$ covers Y, simply observe that $d(t_j, t'_j) \leq r_j$ for all $1 \leq j \leq k$ and apply the triangle inequality: any point in D_j must be at distance at most $2r_j$ of t'_j. The cost of this new solution is clearly twice that of $\mathcal{D}$. $\square$

By a double appplication of this lemma, first about an arbitrary p_0 yielding a point $p' = p_i$, then about p_i yielding another $p' = p_j$, it is immediate that any ℓ-centered cover of Y can be transformed into an $\ell_{i,j}$-centered cover whose cost is increased at most four-fold, where $\ell_{i,i}$ is the line joining p_i and p_j. By computing (exactly or approximately) the optimal set of disks for all $O(n^2)$ lines defined by two different points of Y, we conclude:

Theorem 13 *In $O(n^4 \log n)$ time, we can find a collinear set of disks that cover P at cost at most $4OPT$, and in $O(n^3 \log n)$ time, we can find a collinear set of disks that cover P at cost at most $8\sqrt{2}OPT$.*

3.3.2 A PTAS for finding the best line with unconstrained orientation We prove (in the Appendix) that finding the best line with unconstrained orientation and a minimum-cost covering with disks whose centers are on that line admits a PTAS.

Theorem 14 *Let P be a set of n clients in the plane that can be covered by an optimal collinear set of disks at cost OPT, and $\varepsilon > 0$. In $O((n^5/\varepsilon^2)\log n)$ time, we can find a collinear set of disks that cover P at cost at most $(1+\varepsilon)OPT$.*

4 Minimum-Cost Covering Tours

We now consider the minimum cost covering tour (MCCT) problem: Given $k \geq 1$ and a set $Y = \{p_1, \ldots, p_n\}$ of n clients, determine a cover of Y by (at most) k disks centered at $X = \{t_1, \ldots, t_k\}$ with radii r_j and a tour T visiting X, such that the cost $\text{length}(T) + C\sum r_i^\alpha$ is minimized. We refer to the tour T, together with the disks centered on X, as a *covering tour* of Y. Our results are for the case of linear transmission costs ($\alpha = 1$). We first show a weak hardness result, then characterize the solution for $C \leq 4$, and finally give a PTAS for a fixed $C > 4$.

4.1 A hardness result

We prove the NP-hardness of MCCT where C is also part of the input. Note that this does not prove the NP-hardness of MCCT where C is a fixed constant, which is the problem for which we give a PTAS below. Note also that C appears in the run time exponent of that PTAS, and so the PTAS no longer runs in polynomial time if C is not a fixed constant.

Theorem 15 *MCCT with linear cost is NP-hard if the ratio C is part of the input.*

Proof (sketch). We show a reduction from HAMILTON CYCLE IN GRID GRAPHS. Given a set of n points on a grid, we construct an instance of MCCT in which each of the given points is a client. We set C to be larger than $2n$. In the full proof, we show that this grid graph has a Hamiltonian cycle if and only if there is a covering tour T whose cost is at most n. □

4.2 The case $C \leq 4$: The exact solution is a single circle

Theorem 16 *In the plane, with a cost function of $\text{length}(T) + C\sum r_i$ and $C \leq 4$, the minimum-cost solution is to broadcast to all clients from the circumcenter of the client locations and no tour cost.*

The proof rests on the following elementary geometry lemma (whose proof is omitted here).

Lemma 17 *For three points p, q and r in the plane, such that the triangle pqr contains its own circumcenter, the length of a trip from p to q to r and back to p is at least $4r$ where r is the circumradius of the points.*

Proof of Theorem 16. Let $r(X)$ and $r(Y)$ denote the minimum radius of a circle enclosing X or Y, respectively. Let T be a covering tour of Y, $X \subseteq T$ be the set of disk centers and r_j their radii. Finally, let $r_{\max} = \max_j r_j$.

By the triangle inequality, Lemma 17 implies that the $\text{length}(T) \geq 4r(X)$. Since the tour visits all the centers in X and the disks centered at X cover Y, we have $r(Y) \leq r(X) + r_{\max}$. By definition, the cost of T is $\text{length}(T) + C\sum_j r_j$, which by the observation above is at least $4r(X) + C\sum_j r_j \geq 4r(X) + Cr_{\max}$. The assumption $C \leq 4$ then implies that it be at least $C(r(X) + r_{\max}) \geq Cr(Y)$, which is the cost of covering by a single disk with a zero-length tour. □

4.3 The case $C > 4$: A PTAS

We distinguish between two cases for the choice of transmission points: they may either be arbitrary points in the plane (selected by the algorithm) or they may be constrained to lie within a discrete set $\mathcal{T}$ of candidate locations.

The constant C specifies the relative weight associated with the two parts of the cost function – the length of the tour, and the sum of the disk radii. If C is very small ($C \leq 4$), then the solution is to cover the set Y using a single disk (the minimum enclosing disk), and a corresponding tour of length 0 (the singleton point that is the center of the disk). If C is very large, then the priority is to minimize the sum of the radii of the k disks. Thus, the solution is to compute a covering of Y by k disks that minimizes the sum of radii (as in [18]), and then link the resulting disk centers with a traveling salesman tour (TSP). (In the case that $k \geq n$, the disks in the covering will be of radius 0, and the problem becomes that of computing a TSP tour on Y.) Note that our algorithm gives an alternative to the Lev-Tov and Peleg PTAS [18] for coverage alone.

Our algorithm is based on applying the m-guillotine method [19], appropriately adapted to take into account the cost function and coverage constraint.[1] We need several definitions; we largely follow the notation of [19]. Let $G = (V, E)$ be an embedding of a connected planar graph, of total Euclidean edge-length L. Let $\mathcal{D}$ be a set of disks centered at each vertex v of G of radius r_v. We refer to the pair $(G, \mathcal{D})$ as a *covering network* if the union $\cup_{v\in V} D_v$ of the disks covers the clients Y. We can assume without loss of generality that G is restricted to the unit square B, i.e., $\cup_{e\in E}\, e \subset int(B)$.

Our algorithm relies on there being a polynomial-size set of candidate locations for the transmission points that will serve as the vertices of the covering tour we compute. In the case that a set $\mathcal{T}$ of candidate points is given, this is no issue; however, in the case that the transmission points are arbitrary, we appeal to the following grid-rounding lemma (proved in the Appendix).

Lemma 18 *One can perturb any covering network $(G, \mathcal{D})$ to have its vertices all at grid points on a regular grid of spacing $\delta = O(\varepsilon \cdot diam(S)/n)$, while increasing the total cost by at most a factor of $(1+\varepsilon)$.*

An axis-aligned rectangle, $W \subseteq B$, is called a *window*; rectangle W will correspond to a subproblem in a dynamic programming algorithm. An axis-parallel line ℓ that intersects W is called a *cut*.

For a covering network with edge set E and a set of disks $\mathcal{D}$, we say that $(E, \mathcal{D})$ satisfies the *m-guillotine property with respect to window* W if either (1) all clients $Y \subset W$ lie within disks of $\mathcal{D}$ that intersect the boundary of W; or (2) there exists a cut ℓ with certain properties (an *m-good cut with respect to* W) that splits W into W_1 and W_2, and $(E, \mathcal{D})$ recursively satisfies the m-guillotine property with respect to both W_1 and W_2. Due to the lack of space, we cannot give the full definition of an m-good cut (see the Appendix).

A crux of the method is a structural theorem which shows how to convert any covering network $(G, \mathcal{D})$ into another covering network $(G', \mathcal{D}')$, such that the new graph G' satisfies the m-guillotine property, and that the total cost of the new instance $(G', \mathcal{D}')$ is at most $O((L+CR)/m)$ larger than the original instance $(G, \mathcal{D})$, where L is the total edge length of G and R the sum of the radii of $\mathcal{D}$. The construction is recursive: at each stage, we show that there exists a cut with respect to the current window W (which initially is the unit square B), such that we can afford (by means of a charging scheme) to add short horizontal/vertical edges in order to satisfy the m-guillotine property, without increasing the total edge length too much..

We then apply a dynamic programming algorithm, running in $O(n^{O(m)})$ time, to compute a minimum-cost covering network having a prescribed set of properties: (1) it satisfies the m-guillotine property (with respect to B), which is necessary for the dynamic program to have the claimed efficiency; (2) its disks cover the clients Y ; and (3) its edge set contains an Eulerian subgraph. This third condition allows us to extract a tour in the end. In the proof of the following theorem (see Appendix), we give the details of the dynamic programming algorithm that yields:

Theorem 19 *The min-cost covering tour problem has a PTAS.*

References

[1] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.

[2] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comput. Geom.: Theory Appl.*, 17(1–2):25–50, 2000.

[3] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Math.*, 55(3):197–218, 1994.

[4] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proc. 30th Annu. ACM Symp. Theory Computing*, pages 106–113, 1998.

[5] C. Bajaj. Proving geometric algorithm non-solvability: An application of factoring polynomials. *J. Symbol. Comput.*, 2(1):99–102, 1986.

[6] C. Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Comput. Geom.*, 3:177–191, 1988.

[7] V. Bilò, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *Proc. 13th European Symp. Algorithms*, volume 3669 of *LNCS*, pages 460–471, 2005.

[8] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Comput. Geom.*, 14(4):463–479, 1995.

[9] M. Charikar and R. Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Computer Systems Sci.*, 68(2):417–441, 2004.

[10] A. E. F. Clementi, P. Penna, and R. Silvestri. On the power assignment problem in radio networks. Technical Report TR00-054, Electronic Colloquium on Computational Complexity, 2000.

[11] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *J. Algorithms*, 48(1):135–159, 2003.

[12] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation scheme for geometric graphs. *SIAM J. Computing*, 34(6):1302–1323, 2005.

[1] The “m” in this section refers to a parameter, which is $O(1/\varepsilon)$, not the number of servers.

[13] S. P. Fekete, R. Klein, and A. Nüchter. Searching with an autonomous robot. In *Proc. 20th ACM Annu. Symp. Comput. Geom.*, pages 449–450, 2004. Video available at `http://compgeom.poly.edu/acmvideos/socg04video/`.

[14] S. P. Fekete, R. Klein, and A. Nüchter. Online searching with an autonomous robot. In M. E. *et al.*, editor, *Algorithmic Foundations of Robotics VI*, volume 17 of *Tracts in Advanced Robotics*, pages 139–154. Springer, Berlin, Germany, 2005.

[15] T. F. Gonzalez. Covering a set of points in multidimensional space. *Inf. Proc. Letters*, 40(4):181–188, 1991.

[16] J. Hershberger. Minimizing the sum of diameters efficiently. *Comput. Geom.: Theory Appl.*, 2(2):111–118, 1992.

[17] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

[18] N. Lev-Tov and D. Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005.

[19] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, *k*-MST, and related problems. *SIAM J. Computing*, 28(4):1298–1309, 1999.

[20] K. Pahlavan and A. H. Levesque. *Wireless information networks*, volume 001. Wiley, New York, NY, 2nd edition, 2005.

Appendix

4.4 Sketch of Theorem 3

Let I be an instance of PLANAR 3SAT, and let G_I be the corresponding variable-clause incidence graph. After choosing a suitable layout of this planar graph, resulting in integer variables with coordinates bounded by a polynomial in the size of G_I for all vertices and edges, we replace each the vertex representing any particular variable by a closed loop, using the basic idea shown in the left of Figure 4; this allows two fundamentally different ways of covering those points cheaply (using the "odd" or the "even" circles), representing the two truth assignments. For each edge from a vertex to a variable, we attach a similar chain of points that connects the variable loop to the clause gadget; the parity of covering a variable loop necessarily assigns a parity to all incident chains. Note that choosing sufficiently fine chains guarantees that no large circles can be used, as the overall weight of all circles in a cheap solution will be less than 1. (It is straightforward to see that for any fixed $\alpha > 1$, this can be achieved by choosing coordinates that are polynomial in the size of G_I, with the exponent being $O(1/(\alpha-1))$.)

For the clauses choose a hexagonal arrangement as shown in the right of Figure 4: There is one central point that must be covered somehow; again, $\alpha > 1$ guarantees that it is cheaper to do this from a nearby transmission point, rather than increasing the size of a circle belonging to a chain gadget.

Now it is straightforward to see that there is a cheap cover, using only the forced circles, iff the truth assignment corresponding to the covering of variabe loops assures that each clause has at least one satisfying variable.

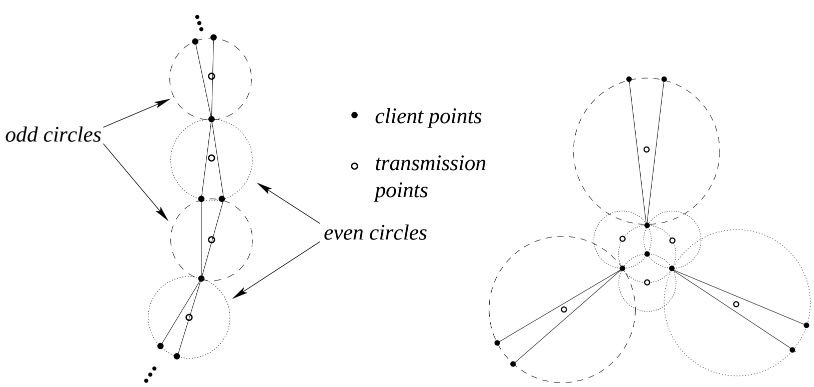

Figure 4. (Left) The switch structure of a variable gadget. Note how there are two fundamentally different ways to cover all points cheaply. (Right) The structure of a clause gadget. One small circle is needed for picking up the client point at the center of the gadget.

4.5 Proof of Theorem 9

As we process points p_i using SGG, attribute to each point p_i a line segment s_i alongp L as follows. If processing p_i resulted in the placement of a square S_i centered at the projection of p_i in L then attribute to p_i the projection on L of a horizontal edge of S_i (Case 1). If, on the other hand, processing of p_i resulted in the growing of a prior square S_j to just capture p_i, attribute to p_i the projection on L of the portion of the horizontal edge of the expanded S_j needed to capture p_i (Case 2). (This amount is at most the distance of p_i to L since otherwise p_i would have been fallen into case 1.) We must show that the lengths of the segments is no more than twice the edge lengths of squares in OPT.

It suffices to show that for any square S in OPT, the segments s_i associated with points $p_i \in S$ processed by SGG cannot have total edge length which exceeds twice the edge length s of S.

To see this observe that the sum of the lengths of those s_i lying completely inside S does not exceed s since they are nonoverlapping. In addition, each of the parts of the at most two segments protruding from S can have length at most $s/2$, in case 1 for the same reason as in the SG algorithm, in case 2 since the total length of the segment is at most $s/2$.

In order to make SGG efficient, we proceed as in SG. In addition, we maintain a balanced binary search tree containing the x-coordinates of the vertical sides of the squares already constructed. For each new point p_i to be processed we locate its x-coordinate within this structure to obtain its neighboring squares and to decide whether case 1 or case 2 applies. This can be done in time $O(\log n)$ just as adding a new square in case 1 or updating an existing square in case 2. Removing points covered by the new or updated square is done as in SG, so that the total runtime remains $O(n \log n)$. □

4.6 Proof of Theorem 10

The following definitions and facts can be found in a standard abstract algebra reference, for instance by Rotman [Rot02].A polynomial with rational coefficients is *solvable by radicals* if its roots can be expressed using rational numbers, the field operations, and taking kth roots. The *splitting field* of a polynomial $f(x)$ over the field of rationals $\mathbb{Q}$ is the smallest subfield of the complex numbers containing all of the roots of $f(x)$. The *Galois group* of a polynomial $f(x)$ with respect to the coefficient field $\mathbb{Q}$ is the group of automorphisms of the splitting field that leave $\mathbb{Q}$ fixed. If the Galois group of $f(x)$ over $\mathbb{Q}$ is a symmetric group on five or more elements, then $f(x)$ is not solvable by radicals over $\mathbb{Q}$.

Consider the following set of points:$\{(3,4),(-3,-2),(102,2),(98,-2),(200,-2)\}$. By exhaustive case analysis, we can show that the optimal solution must consist of one circle through the first two points, a second circle through the next two points, and a third circle touching the last point, and the optimal horizontal line must lie in the range $-2 \leq y \leq 2$. For a given value of y in this range, the cost of the best cover is

$$c(y) = \sqrt{2(y-1)^2+18} + \sqrt{2y^2+8} + (2-y).$$

Therefore, in order to find the best horizontal line, we must minimize $c(y)$. Setting the derivative to zero, we obtain the equation

$$c'(y) = \frac{2(y-1)}{\sqrt{2(y-1)^2+18}} + \frac{2y}{\sqrt{2y^2+8}} - 1 = 0.$$

We easily verify that $c''(y)$ is always positive. The minimum value $c(y) \approx 8.3327196$ is attained at $y \approx 1.4024709$, which is a root of the following polynomial:

$$f(y) \quad = \quad 1024+512y-1600y^2+1536y^3-960y^4+368y^5-172y^6+28y^7-7y^8.$$

Using the computational system GAP [GAP],we compute that the Galois group of $f(y)$ is the symmetric group S_8, so the polynomial is not solvable by radicals.

[GAP] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4*, 2005. `http://www.gap-system.org`.

[Rot02] J. Rotman. *Advanced Modern Algebra*. Prentice Hall, 2002.

Proof of Theorem 14

Let H be a strip of minimal width h that contains P. Using a rotating-caliper approach, H can be computed in $O(n \log n)$ time, cf. [Tou83].If $h = 0$, we can conclude that $OPT = 0$, and we are done.

Otherwise, we can assume wlog that H is horizontal. Let w denote the width of the smallest enclosing axis-parallel rectangle R of P. The height of R is h, and $h \leq w$. Moreover $h/2 \leq OPT \leq \min(w/\sqrt{2}, nh/2)$.

Let APP be the cost of the solution computed in $O(n^4 \log n)$ time according to Theorem 13, $d = \frac{\varepsilon APP}{wn}$, and $\theta = \tan^{-1}(d)$.

Assume an optimal collinear solution $\mathcal{S}^* = \{d(t_1^*, r_1^*), \ldots, d(t_k^*, r_k^*)\}$ lies on the line ℓ^*. Let z be an intersection point of ℓ^* with the boundary of R. For all $p \in P$ we have that $\|z - p\| \leq \sqrt{2}w$.

We now distinguish two cases:

Case 1. $\mathbf{w \leq 2nh}$: For $1 \leq i \leq \lceil \frac{\pi}{2\theta} \rceil$ let ℓ_i be the line through the origin that forms an angle of $i\theta$ with the x-axis.

The number of these lines is $N = O(1/\theta) = O(1/\tan^{-1}(d))$. Note that for $n \geq 4\sqrt{2}\varepsilon$ we have that $d \leq 1/2$, and therefore $\tan^{-1}(d) > d/2$, so $N = O(1/d) = O(\frac{wn}{\varepsilon APP}) = O(\frac{wn}{\varepsilon OPT})$. Since $w \leq 2nh$ and $h/2 \leq OPT$ we get $N = O(\frac{wn}{\varepsilon OPT}) = O(\frac{n^2 h}{\varepsilon OPT}) = O(\frac{n^2}{\varepsilon})$.

Let OPT_i denote the cost of an optimal ℓ-collinear set of disks that cover P, where ℓ is a line that is parallel to ℓ_i. We claim that $\min_i OPT_i \leq (1+4\sqrt{2}\varepsilon)OPT$.

To see this, observe that there is some ℓ_i such that the smallest angle β_i between ℓ_i and ℓ^* is at most θ. If we rotate ℓ^* by β_i (in the right direction) around z, it becomes parallel to ℓ_i. Let $\hat{\ell}$ denote this rotated line. When rotating ℓ^* we also rotate the midpoints $t_1^*, \ldots, t_k^*$ of $\mathcal{S}^*$. By $\hat{t}_1, \ldots, \hat{t}_k$ we denote their final position on $\hat{\ell}$.

Since for all $p \in P$, $\|p - z\| \leq \sqrt{2}w$, the distance of p to $\hat{\ell}$ is at most $\Delta = \sqrt{2}w \tan\theta = \sqrt{2}wd = \sqrt{2}\frac{\varepsilon APP}{n}$ larger than its distance to ℓ^*.

Thus, the set of disks $\hat{\mathcal{S}} = \{d(\hat{t}_1, r_1^* + \Delta), \ldots, d(\hat{t}_k, r_k^* + \Delta)\}$, on $\hat{\ell}$ covers P at a cost of at most $\sum_{i=1}^{k}(r_i^* + \Delta) \leq OPT + \sum_{i=1}^{k}\sqrt{2}\frac{\varepsilon APP}{n} \leq (1 + 4\sqrt{2}\varepsilon)OPT$.

We can compute a $(1+\varepsilon)$-approximation APP_i to OPT_i in $O(n^3 \log n)$ time. We then get that $\min_i APP_i \leq (1 + 5\sqrt{2}\varepsilon)OPT$, and the total running time of this procedure is $O(Nn^3 \log n) = O((n^5/\varepsilon^2)\log n)$.

Case 2. $\mathbf{w > 2nh}$: We will take a slightly different approach here. Let R' be the axis-parallel rectangle obtained from R by moving the left and right side of R inwards by $nh/2$. Clearly the height of R' is h, and its width is $w' = w - nh$. Let S denote the lower left corner of R' and T denote the upper right corner of R'.

If the absolute value of the slope of a line ℓ is larger than $M = h/w'$ (i.e., the aspect-ratio of R') the line ℓ either intersects the top and the bottom edge of R', or S and T lie on the same side of ℓ. In any case, there is at least one point of P that has distance larger than $nh/2$ to ℓ.

Since $OPT < nh/2$, we can therefore conclude that the absolute value of the slope of ℓ^* can be at most M. With $w > 2nh$ it follows that $M < 2h/w \leq 2$.

Let $\lambda = \tan^{-1}(M)$ and for $1 \leq i \leq \lceil \frac{\lambda}{\theta} \rceil$ let ℓ_i be the line through the origin that forms an angle of $i\theta$ with the x-axis.

The number of these lines is $N = O(\frac{\lambda}{\theta}) = O(\frac{\tan^{-1}(M)}{\tan^{-1}(d)})$. Note that as before $\tan^{-1}(d) > d/2$, and since $M \leq 2$, $\tan^{-1}(M) \leq M$, so $N = O(M/d) = O(\frac{hwn}{w\varepsilon APP}) = O(\frac{hn}{\varepsilon OPT})$. Since $h/2 \leq OPT$ we get $N = O(\frac{n}{\varepsilon})$.

With the above reasoning, we can argue as before that there is some ℓ_i such that the smallest angle β_i between ℓ_i and ℓ^* is at most θ. If we rotate ℓ^* by β_i (in the right direction) around z, it becomes parallel to ℓ_i. Let $\hat{\ell}$ denote this rotated line.

Repeating the previous argumentation, we get a solution on $\hat{\ell}$ that covers P at a cost of at most $(1 + 4\sqrt{2}\varepsilon)OPT$.

Again, we can compute a $(1+\varepsilon)$-approximation APP_i to OPT_i in $O(n^3 \log n)$ time and get that $\min_i APP_i \leq (1 + 5\sqrt{2}\varepsilon)OPT$, and the total running time of this procedure is $O(Nn^3 \log n) = O((n^4/\varepsilon)\log n)$. $\square$

[Tou83] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, pages A10.02/1–4, 1983.

Proof Lemma 18

By making a detour for each vertex of the network, to the closest point on a grid of spacing $\delta = O(\varepsilon \cdot diam(S)/n)$, we add length at most $n\delta = O(\varepsilon \cdot diam(S))$, since there are at most n vertices of the network (otherwise, the cost of the network can be improved by deleting some vertices). Now, we can shift the disks of $\mathcal{D}$ to be centered at the corresponding grid points, and slightly enlarge them (by radius δ), so that they still cover all of Y. The total increase in cost is then $O(n\delta(1+C)) = O(\varepsilon \cdot diam(S))$. Since $\Omega(diam(S))$ is a lower bound on the cost of an optimal solution, we see that the grid-rounded solution is within a factor $(1+\varepsilon)$ of optimal.

Complete details of the PTAS for the MCCT problem

An axis-aligned rectangle, $W \subseteq B$, is called a *window*; rectangle W will correspond to a subproblem in a dynamic programming algorithm. An axis-parallel line ℓ that intersects W is called a *cut*.

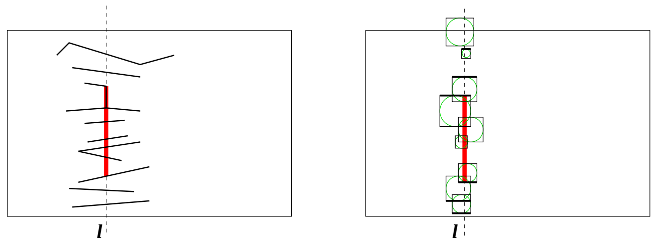

Figure 5. Left: The 3-span, $\sigma_3(\ell)$ of ℓ with respect to the window $W \subset B$ is highlighted with a thick shaded vertical segment. Right: The 3-disk-span, $\sigma_{3,\mathcal{D}}(\ell)$ of ℓ with respect to the window $W \subset B$ is highlighted with a thick shaded vertical segment.

For a positive integer m, we define the *m-span*, $\sigma_m(\ell)$, of ℓ with respect to W, as follows. For each edge e of E, $e \cap \ell \cap \mathrm{int}(W)$ is either empty, a point, or a subsegment if $e \subseteq \ell$ and $e \cap \mathrm{int}(W) \neq \emptyset$. Let $p_1, \ldots, p_\xi$ denote all these points and endpoints of subsegments (with multiplicities), in order of decreasing y-coordinate along ℓ. If $\xi > 2(m-1)$, then $\sigma_m(\ell)$ is the line segment $p_m p_{\xi-m+1}$, else $\sigma_m(\ell) = \emptyset$. Note that $\sigma_m(\ell)$ may be of zero length in case $p_m = p_{\xi-m+1}$.

We also define the *m-disk span*, $\sigma_{m,\mathcal{D}}(\ell)$ of ℓ with respect to W, as follows. First, we surround each (circular) disk D_v with an axis-aligned square B_v of side length $2r_v$, centered at v. Let $\mathcal{B} = \{B_v : v \in V\}$, $\mathcal{B} \cap W = \{B_v : B_v \cap W \neq \emptyset\}$, and $\xi_{\mathcal{B}} = |\mathcal{B} \cap W|$. Starting from the top (resp. bottom) endpoint of $\ell \cap W$ and going downwards (resp. upwards), let u_m (resp. b_m) be the point where ℓ crosses the mth top (resp. bottom) edge of a bounding box, if such a point exists. If both u_m and b_m exist and if u_m is above b_m, then $\sigma_{m,\mathcal{D}}(\ell)$ is the vertical line segment $u_m b_m$, else $\sigma_{m,\mathcal{D}}(\ell) = \emptyset$. (The definition is similar if the cut ℓ is horizontal.)

We say that a cut ℓ is an *m-good cut with respect to W* if $\sigma_m(\ell) \subseteq E$ and $\sigma_{m,\mathcal{D}}(\ell)$ is either empty or lies fully within a single disk of $\mathcal{D}$. In particular, if $\xi \leq 2(m-1)$ and $\xi_{\mathcal{B}} \leq 2m$, then ℓ is trivially an m-good cut, since both the m-span and the m-disk-span are empty in this case.

For a covering network with edge set E and a set of disks $\mathcal{D}$, we say that $(E, \mathcal{D})$ satisfies the *m-guillotine property with respect to window W* if either (1) all clients $Y \subset W$ lie within disks of $\mathcal{D}$ that intersect the boundary of W; or (2) there exists an m-good cut ℓ with respect to W that splits W into W_1 and W_2, and $(E, \mathcal{D})$ recursively satisfies the m-guillotine property with respect to both W_1 and W_2.

Theorem 20 *Let G be an embedded connected planar graph on vertices V, with edge set E, of total length L, and a set $\mathcal{D}$ of $k = |V|$ disks, of total radii R, centered at vertices of V that cover a given set Y of clients. Assume that E and $\mathcal{D}$ are contained in the unit square B. Then, for any positive integer m, there exists a planar graph G', together with a set $\mathcal{D}'$ of at most k disks centered on the vertices V' of G', that satisfies the m-guillotine property with respect to B, has an edge set $E' \supseteq E$ of length $L' \leq \left(1 + \frac{\sqrt{2}}{m}\right) L$ and has vertices V' either at points of an input set $\mathcal{T}$ or at points of a regular grid of spacing $\delta = O(\varepsilon \cdot diam(S)/n)$.*

Proof. We show how to convert G into a new graph G' by adding to E a new set of horizontal/vertical edges whose total length is "small", and we convert $\mathcal{D}$ into a new set, $\mathcal{D}'$, of at most k covering disks centered at the vertices of G' (which are a subset of $\mathcal{T}$ or of the regular δ-grid), so that the total cost of the new instance $(E', \mathcal{D}')$ is at most $O((L + CR)/m)$ longer than the original instance $(E, \mathcal{D})$.

The construction is recursive: at each stage, we show that there exists a cut, ℓ, with respect to the current window W (which initially is the unit square B), such that we can "afford" (by means of a charging scheme) to add both the m-span and the m-disk-span to E.

A point p on a cut ℓ is *m-dark with respect to ℓ and W* if, along $\ell^\perp \cap int(W)$, there are at least m edges of E intersected by $\ell^\perp$ on each side of p, where $\ell^\perp$ is the line perpendicular to ℓ passing through p. We say that a subsegment of ℓ is *m-dark* (with respect to W) if all points of the segment are m-dark with respect to ℓ and W. The important property of m-dark points along ℓ is the following: Assume, without loss of generality, that ℓ is horizontal. We consider any line segment that lies along an edge of E to have a *top* side and a *bottom* side; the top is the side that can be seen from above, from a point with $y = +\infty$. Then, if all points on subsegment pq of ℓ are m-dark, we can charge the length of pq off to the bottoms of the first m subsegments, $E^+ \subseteq E$, of edges that lie above pq, and the tops of the first m subsegments, $E^- \subseteq E$, of edges that lie below pq, since we know that there are at least m edges "blocking" pq from the top/bottom of W. We charge pq's length half to E^+, charging each of the m levels of E^+ *from below*, with $\frac{1}{2m}$ units of charge, and half to E^-, charging each of the m levels of E^- *from above*, with $\frac{1}{2m}$ units of charge. We refer to this type of charge as the "red" charge.

A point p on a cut ℓ is *m-disk-dark with respect to ℓ and W* if, along $\ell^\perp \cap int(W)$, there are at least m disks of $\mathcal{D}$ whose top edges of their bounding box lie above p in W *and* there are at least m disks of $\mathcal{D}$ whose bottom edges of their bounding box lie below p in W. We say that a subsegment of ℓ is *m-disk-dark* with respect to W if all points of the segment are m-disk-dark with respect to ℓ and W. The *chargeable* length within W of a cut ℓ is defined to be the sum of the lengths of its m-dark portion and C times its m-disk-dark portion. Refer to Figure 6.

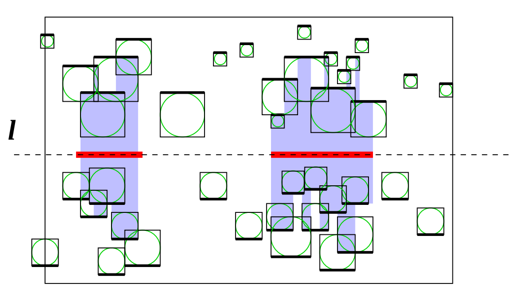

Figure 6. Definition of m-disk-dark: Here, the points that are 2-disk-dark with respect to ℓ are those two highlighted subsegments of ℓ that lie within the shaded regions, which comprise the set of points $R_x^{(2,\mathcal{D})}$ of W that are 2-disk-dark with respect to horizontal cuts.

The important property of m-disk-dark points along horizontal ℓ is the following: If all points on subsegment pq of ℓ are m-disk-dark, then we can charge the length of pq off to the tops of the bounding boxes of the first m disks that lie above pq, and the bottoms of the bounding boxes of the first m disks that lie below pq, since we know that there are at least m disks "blocking" pq from the top/bottom of W. We charge pq's length half upwards, charging the tops of each of the m "levels" of bounding boxes of disks with $\frac{1}{2m}$ units of charge, and half downwards, charging the bottoms of each of the m "levels" of bounding boxes of disks with $\frac{1}{2m}$ units of charge. We refer to this type of charge as the "blue" charge.

We say that a cut ℓ is *favorable* if its chargeable length within W is *at least as large as* the sum of the lengths of the m-span and C times the m-disk-span. The existence of a favorable cut is guaranteed by the following key lemma, whose proof (in the Appendix) is similar to that of the key lemma in [11, 19]:

Lemma 21 *For any G, with disk set $\mathcal{D}$, and any window W, there is a favorable cut.*

Proof. We show that there must be a favorable cut that is either horizontal or vertical.

Let $f(x)$ denote the "cost" of the vertical line, ℓ_x, passing through the point $(x,0)$, where "cost" means the sum of the lengths of the m-span and the m-disk-span for ℓ_x. Then,

$$f(x) = |\sigma_m(\ell_x)| + C|\sigma_{m,\mathcal{D}}(\ell_x)|.$$

Thus,

$$A_x = \int_0^1 f(x)dx = A_x^{(m)} + C\cdot A_x^{(m,\mathcal{D})} = \int_0^1 |\sigma_m(\ell_x)|dx + C\cdot\int_0^1 |\sigma_{m,\mathcal{D}}(\ell_x)|dx,$$

where $A_x^{(m)} = \int_0^1 |\sigma_m(\ell_x)|dx$ is the area of the x-monotone region $R_x^{(m)}$ of points of B that are m-dark with respect to horizontal cuts, and $A_x^{(m,\mathcal{D})} = \int_0^1 |\sigma_{m,\mathcal{D}}(\ell_x)|dx$ is the area of the x-monotone region $R_x^{(m,\mathcal{D})}$ of points of B that are m-disk-dark with respect to horizontal cuts. Refer to Figure 6. Similarly, define $g(y)$ to be the cost of the horizontal line through y, and let $A_y = \int_0^1 g(y)dy$.

Assume, without loss of generality, that $A_x \geq A_y$. We claim that there exists a horizontal favorable cut; i.e., we claim that there exists a horizontal cut, ℓ, such that its chargeable length is at least as large as the cost of ℓ, meaning that the length of its m-dark portion plus its m-disk-dark portion is at least $|\sigma_m(\ell)| + |\sigma_{m,\mathcal{D}}(\ell)|$. To see this, note that A_x can be computed by switching the order of integration, "slicing" the regions $R_x^{(m)}$ and $R_x^{(m,\mathcal{D})}$ horizontally, rather than vertically; i.e., $A_x = \int_0^1 h(y)dy = \int_0^1 h_m(y)dy + C\cdot\int_0^1 h_{m,\mathcal{D}}(y)dy$, where $h_m(y)$ is the m-dark length of the horizontal line through y, $h_{m,\mathcal{D}}(y)$ is the length of the intersection of $R_x^{(m,\mathcal{D})}$ with a horizontal line through y, and $h(y)$ is the chargeable length of the horizontal line through y. In other words, $h_m(y)$ (resp., $h_{m,\mathcal{D}}(y)$) is the length of the m-dark (resp., m-disk-dark) portion of the horizontal line through y. Thus, since $A_x \geq A_y$, we get that $\int_0^1 h(y)dy \geq \int_0^1 g(y)dy \geq 0$. Thus, it cannot be that for all values of $y \in [0,1]$, $h(y) < g(y)$, so there exists a $y = y^*$ for which $h(y^*) \geq g(y^*)$. The horizontal line through this y^* is a cut satisfying the claim of the lemma.

If, instead, we had $A_x \leq A_y$, then we would get a *vertical* cut satisfying the claim. □

Now that we know there must be a favorable cut, ℓ, we can charge off the cost of the m-span and the m-disk-span of ℓ, making "red" charge on the bottoms (resp., tops) of segments of E that lie above (resp., below) m-dark points of ℓ, and making "blue" charge on the tops (resp., bottoms) of bounding boxes of disks that lie above (resp., below) m-disk-dark points of ℓ. We then recurse on each side of the cut, in the two new windows.

After a portion of E has been charged *red* on one side, due to a cut ℓ, it will be within m levels of the boundary of the windows on either side of ℓ, and, hence, within m levels of the boundary of any future windows, found deeper in the recursion, that contain the portion. Thus, no portion of E will ever be charged *red* more than once from each side, in each of the two directions, horizontal or vertical, so no portion of E will ever pay more than $\sqrt{2}/m$ times its length in red charge. It is important to note that we are always charging red portions of the original edge set E: the new edges added are never themselves charged, since they lie on window boundaries and cannot therefore serve to make a portion of some future cut m-dark. We charge at the rate of $\frac{1}{2m}$ per unit length of the perimeter of the segment's axis-aligned bounding box, and the worst case is achieved for a segment of slope ± 1. Thus, the total red charge is at most $\frac{\sqrt{2}}{m}L$. Similarly, no side of a bounding box of a disk of $\mathcal{D}$ will ever be charged *blue* more than once. Since we do blue charging at the rate of $\frac{1}{2m}$ per unit length of the bounding box, we get a total blue charge of at most $\frac{8R}{m}$.

Overall, then, the total increase in cost caused by adding the m-spans and m-disk-spans along favorable cuts is bounded by $\frac{\sqrt{2}}{m}L + C\cdot\frac{8}{m}R$.

Thus, we can afford to add to E the vertical/horizontal segments that are the m-spans of the favorable cuts. Also, we can afford to replace the set of disks of $\mathcal{D}$ that intersect the m-disk-span, $u_m b_m$, with a *single* disk, centered at the midpoint of $u_m b_m$, of radius $|u_m b_m|$. This disk is large enough to contain the m-disk-span $u_m b_m$ and is also large enough to perform all of the covering that was done previously by the disks pierced by $u_m b_m$, which shows that:

Lemma 22 *The single disk of radius $|u_m b_m|$ centered at the midpoint of $u_m b_m$ is large enough to cover all of the region covered by the disks of $\mathcal{D}$ whose bounding box is pierced (from top to bottom) by $u_m b_m$.*

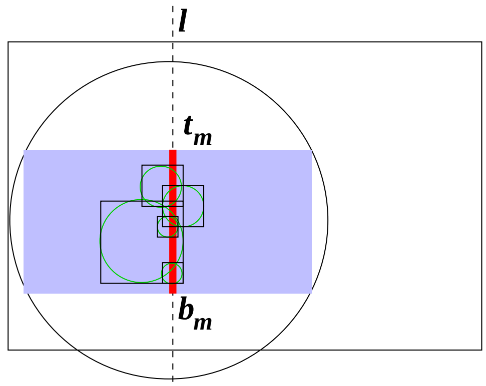

Figure 7. Proof of Lemma.

Furthermore, by adding length at most $O(|u_m b_m|)$ we can also afford to add to E the length of a detour that connects the midpoint of $u_m b_m$ to the center of one of the disks of $\mathcal{D}$ that is pierced by $u_m b_m$ (since pierced disks have radius at most $|u_m b_m|/2$).

Our goal is to obtain a succinct representation of the set of disks that straddle the boundary of a window; indeed, we end up with only a constant-size description, since we replace the possibly many disks intersecting the m-disk-span with a single disk that does at least as much coverage of Y. This allows us to specify succinctly which clients are required to be covered within a subproblem. □

In the proof of the following theorem, we give details of the dynamic programming algorithm that yields:

Theorem 23 *The min-cost covering tour problem has a PTAS.*

Proof. Consider an optimal covering tour, OPT, of length L^*, whose associated disk set $\mathcal{D}^*$ has total sum of radii R^*. The cost of OPT is $L^* + C \cdot R^*$. Now, OPT is a simple polygon. If we are in the case of arbitrary transmission points, we can (by Lemma 18) perturb the vertices of OPT (and slightly grow the disks of $\mathcal{D}^*$) so that each lies at a grid point in a grid of resolution $\delta = O(\varepsilon \cdot diam(S)/n)$; otherwise, we know the vertices of OPT lie among the m input candidate transmission points. Theorem 20 implies that we can convert OPT into a covering network, $(E', \mathcal{D}')$, having the m-guillotine property, while not increasing the total cost by too much. In particular, the cost of OPT' is at most

$$\left(1+\frac{\sqrt{2}}{m}\right)L^* + C\cdot\left(1+\frac{8}{m}\right)R^* \leq (1+O(1/m))(L^* + C\cdot R^*).$$

We now apply a dynamic programming algorithm, running in $O(n^{O(m)})$ time, to compute a minimum-cost covering network having a prescribed set of properties: (1) it satisfies the m-guillotine property, which is necessary for the dynamic program to have the claimed efficiency; (2) its disks cover the clients Y; and (3) its edge set contains an Eulerian subgraph. This third condition allows us to extract a tour in the end. We only outline here the dynamic programming algorithm; the details are very similar to those of [19], with the modification to account for the m-disk-span and the coverage constraints.

A subproblem is defined by a rectangle W whose coordinates are among those of the δ-grid points or of the input set $\mathcal{T}$ of candidate transmission points, together with a constant amount ($O(m)$) of information about how the solution to the subproblem interacts across the boundary of W with the solution outside of W. This information includes the following:

(a) For each of the four sides of W, we specify a "bridge" segment and at most $2m$ other segments with endpoints among $\mathcal{G}$ that cross the side; this is done exactly as in the case of the Euclidean TSP on points, as in [19].

(b) For each of the four sides of W, we specify a "disk bridge" disk corresponding to the disk centered on the m-disk-span, and, we specify each of at most $2m$ disks of $\mathcal{D}$ that are not intersected by the disk bridge segment, but are intersecting the boundary of W. These $O(m)$ disks make up the disk boundary information; all points of Y within W that are *not* already covered by these disks are required to be covered by the subproblem.

(c) We specify a required “connection pattern” within W. In particular, we indicate which subsets of the $O(m)$ specified edges crossing the boundary of W are required to be connected within W. This, again, is done exactly as is detailed for the Euclidean TSP on point sets in [19].

(d) We specify the number, k', of disks that are budgeted to the subproblem.

The dynamic programming algorithm optimizes over all possible cuts, and all possible choices of disk information (using $O(m)$ disks) along the cut, and all possible partitions of the budget k' into the subproblems on each side of the cut.

In order to end up with a network having an Eulerian subgraph (so we can extract easily a tour), we use the same trick as done in [19]: we “double” the bridge segments, and then require that the number of connections on each side of a bridge segment satisfy a parity condition. Exactly as in [19], this allows us to extract a tour from the network that results from the dynamic programming algorithm, which gives a minimum-cost covering network that obeys the specified conditions.

The result is that in polynomial time ($O(n^{O(m)})$) one can compute a minimum-cost covering network, from a special class of such networks (with disk sets), and this network’s disks cover all of Y using at most k disks. Theorem 20 guarantees that the cost of the resulting covering network is very close, within factor $1+O(1/m)$, to the cost, $L^* + C \cdot R^*$, of an optimal solution. Thus, once we extract a tour from the Eulerian subgraph, we have the desired $(1+\varepsilon)$-approximation solution, where $\varepsilon = O(1/m)$. □