A Nearly Linear-Time Approximation Scheme for the Euclidean kappa-median Problem

Stavros G. Kolliopoulos, Satish Rao

Published on: 16 Jul 1999 - European Symposium on Algorithms

Topics: Euclidean distance, Euclidean shortest path, Euclidean distance matrix, Closest pair of points problem and Euclidean domain

A nearly linear-time approximation scheme for the Euclidean k-median problem

Stavros G. Kolliopoulos[*] Satish Rao[†]

October 30, 2006

Abstract

We provide a randomized approximation scheme for the k-median problem when the input points lie in the d-dimensional Euclidean space. The running time is $O(2^{O((\log(1/\varepsilon)/\varepsilon)^{d-1})} n \log^{d+6} n)$, which is nearly linear for any fixed ε and d. Moreover our method provides the first polynomial-time approximation scheme for k-median and uncapacitated facility location instances in d-dimensional Euclidean space for any fixed $d > 2$. Our work extends techniques introduced originally by Arora for the Euclidean TSP. To obtain the improvement we develop a structure theorem to describe hierarchical decomposition of solutions. The theorem is based on an *adaptive decomposition* scheme, which guesses at every level of the hierarchy the structure of the optimal solution and modifies accordingly the parameters of the decomposition. We believe that our methodology is of independent interest and may find applications to further geometric problems.

Keywords: approximation algorithms, approximation schemes, k-median, facility location, Euclidean space, linear time. *AMS subject classifications:* 68Q25, 90B10, 90B12, 90B35.

Running head: Nearly linear-time approximation scheme

[*]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece (`www.di.uoa.gr/~sgk`).

[†]Computer Science Division, University of California Berkeley, CA 94720, USA (`satishr@cs.berkeley.edu`).

1 Introduction

In the *k-median* problem we are given a set N of n points in a metric space and a positive integer k. The objective is to locate k *medians* (facilities) among the points so that the sum of the distances from each point in N to its closest median is minimized. The version of the problem we study is sometimes called the discrete k-median: the facilities must be located at input points. In the continuous version facilities may be located anywhere in the underlying space. The k-median problem is a well-studied, NP-hard problem which falls into the general class of *clustering* problems: partition a set of points into clusters so that the points within a cluster are close to each other with respect to some appropriate measure. Moreover k-median is closely related to *uncapacitated facility location,* a basic problem in the operations research literature (see, e.g., [13]). In the latter problem except for the set N of points we are given also a cost c_i for *opening* a facility at point i. The objective is to open an unspecified number of facilities at a subset of N so as to minimize the sum of the cost to open the facilities (*facility cost*) plus the cost of assigning each point to the nearest open facility (*service cost*). In this paper we provide a fast approximation scheme for the problem when the input lies in a Euclidean space.

A *ρ-approximation algorithm* for a minimization problem, $\rho > 1$, computes in time polynomial in the input size a feasible solution of cost at most ρ times the optimum. An *approximation scheme* computes for any fixed $\varepsilon > 0$, a $(1+\varepsilon)$-approximate feasible solution in time polynomial in the input size and $1/\varepsilon$.

1.1 Previous Work

The succession of results for k-median is as follows. Lin and Vitter [23] used their filtering technique to obtain a solution of cost at most $(1+\varepsilon)$ times the optimum but using $(1+1/\varepsilon)(\ln n+1)k$ medians. They later refined their technique to obtain a solution of cost $2(1+\varepsilon)$ while using at most $(1+1/\varepsilon)k$ medians [22]. The first non-trivial approximation algorithm that achieves feasibility as well, i.e. uses k medians, combined the powerful randomized algorithm by Bartal for approximation of metric spaces by trees [5, 6] with an approximation algorithm by Hochbaum for k-median on trees [19]. The ratio thus achieved is $O(\log n \log\log n)$. This algorithm was subsequently refined and derandomized by Charikar, Chekuri, Goel and Guha [8] to obtain a guarantee of $O(\log k \log\log k)$. Charikar and Guha and independently Tardos and and Shmoys reported the first constant-factor approximations [10]. In contrast, the uncapacitated facility location problem, in which there is no a priori constraint on the number of facilities, seems to be better understood. Shmoys, Tardos and Aardal [27] gave a 3.16 approximation algorithm. This was later improved by Guha and Khuller [16] to 2.408 and to 1.736 by Chudak [12]. After a preliminary, and by now obsolete, abstract[1] of this work appeared in [21] some additional results on k-median and facility location appeared by Charikar and Guha [9], Jain and Vazirani [20] and the local search approach by Arya et al. [4]. Follow-up work on the Euclidean case includes the one by Har-Peled and Mazumdar [18] and work focusing on improving the dependence on the dimension by Chen [11].

In this paper we focus on the case when the underlying metric is Euclidean. Until the work of Arora, Raghavan and Rao [3], this case was not known to be any easier to approximate than a general metric. These authors gave a randomized polynomial-time approximation scheme for k-median when the points lie on the Euclidean plane [3]. For any fixed $\varepsilon > 0$, their algorithm outputs a $(1+\varepsilon)$-approximation with probability $1-o(1)$ and runs in $O(nkn^{O(1/\varepsilon)}\log n)$ time, worst case. For facility location they gave an approximation scheme with running time $O(n^{1+O(1/\varepsilon)}\log n)$. This development followed the breakthrough approximation schemes of Arora [2] for the Traveling

[1]In the conference version [21] we had erroneously claimed a running time of $O(2^{1/\varepsilon^d} n \log n \log k)$.

Salesman Problem and other geometric problems. While the work in [3] used techniques from the TSP approximation scheme, the different structure of the optimal solutions for k-median and TSP necessitated the development of a new structure theorem to hierarchically decompose solutions. We elaborate further on this issue during the exposition of our results in the next paragraph. The dependence of the running time achieved by the methods of Arora, Raghavan and Rao on $1/\varepsilon$ is particularly high. For example, the approximation scheme can be extended to higher-dimension instances but runs in quasi-polynomial time $O(n^{(\log n/\varepsilon)^{d-2}})$ for a set of points in R^d with fixed $d > 2$.

1.2 Results and Techniques

Results. We provide a randomized approximation scheme for k-median on the Euclidean plane. For any fixed $\varepsilon > 0$, our scheme outputs in expectation a $(1+\varepsilon)$-approximate solution, in time

$$O\left(2^{O\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)} n \log^8 n\right)$$

worst case[2]. Our time bound represents a drastic improvement on the result in [3]. For any fixed accuracy ε desired, the dependence of the running time on $1/\varepsilon$ translates to a (large) constant hidden in the near-linear asymptotic bound $O(n \log^8 n)$ compared to the exponent of a term polynomial in n in the bound of Arora, Raghavan and Rao. Moreover, for inputs in R^d, our algorithm extends to yield a running time of

$$O\left(2^{O\left((\frac{\log(1/\varepsilon)}{\varepsilon})^{d-1}\right)} n \log^{d+6} n\right)$$

which yields for the first time a polynomial-time approximation scheme for any fixed $d > 2$. The ideas behind the new k-median algorithm yield also improved, nearly linear-time, approximation schemes for uncapacitated facility location. Our time bounds for the latter problem hold under the assumption that a polynomial in n approximation is available for the value of the service cost. An example of such a case is when all the inter-point distances are polynomially related. We now elaborate on the techniques we use to obtain our results.

Techniques. Our main motivation for improving the running times of [3] was to gain a better understanding of approximation schemes for geometric problems and in particular the issues arising from Euclidean facility location problems. The actual running time we obtain is mainly of theoretical interest. We believe that the main contribution of the present paper lies in the new ideas we introduce to overcome the limitations of the approach employed by Arora, Raghavan and Rao in [3]. To describe these ideas we sketch first some previous developments, starting with the breakthrough results by Arora [2] (see also the work of Mitchell [24] for a different approximation scheme for Euclidean TSP).

A basic building block for Arora's results on TSP [2] was a structure theorem providing insight into how much the cost of an optimal tour could be affected in the following situation. Roughly speaking, the plane is recursively dissected into a collection of rectangles of geometrically decreasing area, represented by a quadtree data structure. For every box in the dissection one places a fixed number, dependent on the desired accuracy ε, of equidistant *portals* on the boundary of the box. The optimal TSP tour can cross between adjacent rectangles any number of times; a *portal-respecting* tour is allowed to cross only at portals. How bad can the cost of a deflected, portal-respecting,

[2]For large ε, the term $\frac{\log(1/\varepsilon)}{\varepsilon}$ should be interpreted throughout the paper as $\frac{\max\{\log(1/\varepsilon),\Theta(1)\}}{\varepsilon}$. We omit the $\Theta(1)$ factor to avoid cumbersome notation.

tour be compared to the optimum? Implicitly, Arora used a charging argument on the edges in an optimal solution to show that the edges could be made to be portal respecting. We sketch now his approach which was made explicit and applied to k-median in [3].

Given a set of at most k open facilities a k-median solution is a set of edges assigning every point to an open facility. A portal-respecting solution is one in which the assignment edges are actually paths that cross rectangle boundaries only at portals. We can assume that the input is surrounded by a rectangle with side length polynomial in n (cf. Section 2). At level i of the dissection, the rectangles at this level with sidelength 2^i are cut by vertical and horizontal lines into rectangles of sidelength 2^{i-1}. The x- and y-coordinates of the dissection are randomly shifted at the beginning, so that the probability that an edge e in a solution is cut at level i is $O(\text{length}(e)/2^i)$. Let m denote the number of portals along the dissection lines. If e is cut at level i it must be deflected through a portal, paying additional cost $O(2^i/m)$. Summing over all the $O(\log n)$ levels of the decomposition, the expected deflection cost of edge e in a portal-respecting solution is at most

$$\sum_{i=1}^{O(\log n)} O(\frac{\text{length}(e)}{2^i}(2^i/m)) \qquad (1)$$

Selecting $m = \Theta(\log n/\varepsilon)$ and applying the dissection to the optimal solution we obtain the existence of a portal-respecting solution of cost $(1+\varepsilon)OPT$. Once the existence has been shown, dynamic programming can be used to compute the best portal-respecting solution. The running time of the dynamic programming contains a $k2^{O(m)}$ term, hence the $kn^{1/\varepsilon}$ term in the overall running time.

Arora additionally used a "patching lemma" argument to show that the TSP could be made to cross each box boundary $O(1/\epsilon)$ times. This yielded an $O(n(\log n)^{O(1/\varepsilon)})$ time algorithm for TSP. (This running time was subsequently improved by Rao and Smith to $O(2^{O(1/\varepsilon^2)} + n\log n)$ [25] while still using $\Theta(\log n/\varepsilon)$ portals). The k-median method did not, however, succumb to a patching lemma argument, thus the running time for the algorithms in [3] remained $O(kn^{O(1/\epsilon)})$.

Our method reduces the number m of portals to $O(\log(1/\epsilon)/\epsilon)$. *That is, we remove the* $\log n$ *factor in the number of portals that appears to be inherent in Arora, Raghavan, and Rao's charging based methods and even in Arora's charging plus patching based methods.*

Adaptive dissection. We outline some of the ideas behind the reduced value for m. The computation in (1) exploits linearity of expectation by showing that the "average" dissection line cutting an edge is short enough. The complicated dependencies among the dissection lines across all $O(\log n)$ levels seem too complicated to reason about directly. On the other hand, when summing the expectations across all levels an $O(\log n)$ factor creeps in, which apparently has to be offset by setting m to $\log n/\varepsilon$. We provide a new structure theorem to characterize the structure of near-optimal solutions. In contrast to previous approaches, given a rectangle at some level in the decomposition, it seems a good idea to choose several possible "cuts" hoping that one of them will hit a small number of segments from the optimum solution. This approach gives rise to the *adaptive dissection* idea, in which the algorithm "guesses" the structure of the part of the solution contained in a given rectangle and tunes accordingly the generation of the sub-rectangles created for the next level of the dissection. In the k-median problem the guess consists of the area of the rectangle which is empty of facilities. Let L be the maximum side length of the sub-rectangle containing facilities. Cutting close to the middle of this sub-rectangle with a line of length L should, in a probabilistic sense, mostly dissect segments from the optimal solution of length $\Omega(L)$, forcing them to deflect by $L/m = \varepsilon L$. A number of complications arise by the fact that a segment may be cut by both horizontal and vertical dissection lines. We note that the cost of "guessing" the empty area is incorporated into the size of the dynamic programming lookup table by trying all possible configurations. Given the preeminence of recursive dissection in approximation schemes for Euclidean problems ([2, 3, 25]) we believe that the adaptive dissection technique is of independent interest and may prove useful in other geometric problems as well.

Although the adaptive dissection technique succeeds in reducing the required number of portals to $\Theta(\log(1/\varepsilon)/\varepsilon)$ and thus asymptotically improve the dependence of the running time on $1/\varepsilon$, the dynamic program has still to enumerate all possible rectangles. Compared to the algorithm in [3] we apparently have to enumerate even more rectangles due to the "guess" for the areas without facilities and some amount of randomization we introduce in the choice of sub-rectangle boundaries. We bound the size of the lookup table by showing that the boundaries of the possible rectangles can be appropriately spaced and still capture the structure of a near-optimal solution.

The outline of the paper is as follows. In Section 2 we give definitions and preprocessing steps. In Section 3 we prove the new structure theorem and obtain the reduced number of portals. In Section 4 we provide a modified structure theorem which yields a small size for the dynamic program table. In Section 5 we present the dynamic program and some extensions of the main result.

2 Preliminaries

An *edge* (u, v) is a line segment connecting input points u and v. Given a selection of open facilities, an *assignment edge* is an edge (u, v) such that exactly one of u or v is an open facility. An *assignment* is a set E of assignment edges such that every point which does not host a facility appears exactly once as an endpoint. For minimum cost every point must of course be assigned to its closest open facility. The *sidelength* of a rectangle with sides parallel to the axes is the length of its largest side. For any two points p, p' their Euclidean distance is denoted by $d(p, p')$.

We assume that the input points are on a unit grid of size polynomial in the number of input points. This assumption is enforced by a preprocessing phase described in [3]; the preprocessing incurs an additive error of $O(1/n^c)$ times the optimal value, for some constant $c > 0$. The assumption is needed to ensure the depth of the recursive dissection is $O(\log n)$. Within the same additive error we can assume that no two input points lie on the same vertical grid line. The latter assumption simplifies the presentation.

In [3] it is shown that if a polynomial-factor approximation is available for the value of the service cost, i.e., a range $[D, n^c D]$ where this value must lie, then the above assumptions on the points can be enforced by a simple plane sweep. An algorithm to compute this range for the k-median problem is the 2-approximation minmax clustering algorithm of Gonzalez [15]. A faster algorithm for the same problem was given by Feder and Greene [14]. The latter runs in $O(n \log k)$ time on the plane. The total preprocessing time for $d = 2$ is $O(n \log n)$. For general dimension d the preprocessing time is $O(dn \log n + kd^d \log(kd^d))$ [14]. A faster 2-approximation algorithm that runs in $O(n + k \log k)$ with high probability was given by Har-Peled in [17].

3 The Structure Theorem

In this section we prove our basic Structure Theorem that shows the existence of approximately optimal solutions with a simple structure. This theorem can yield directly a dynamic-programming algorithm whose running time dependence on ε is no worse than $2^{O(\log(1/\varepsilon)/\varepsilon)}$. Our exposition focuses on 2-dimensional Euclidean instances. It is easy to generalize to d dimensions. Given a set of points N and a set of facilities $F \subset N$, we define the *greedy cost under* F to be the cost of assigning each point to its closest facility. If F is the set of facilities open in the optimal solution, the greedy cost under F is the optimal cost.

We proceed to define a recursive randomized decomposition. The decomposition uses two processes: the Sub-Rectangle process, and the Cut-Rectangle process.

Sub-Rectangle:
Input: a rectangle B containing at least one facility.
Process: Find the minimal rectangle B' containing all the facilities. Let its maximum sidelength be s. Grow the rectangle by $s/3$ in each dimension. We call the grown rectangle B''.
Output: $B_s = B'' \cap B$.

Notice that $B - B_s$ contains no facilities. If $B_s \neq B$, we call B_s a *proper sub-rectangle.*

Cut-Rectangle:
Input: a rectangle B containing at least one facility.
Process: Randomly cut the rectangle into two rectangles with a line that is orthogonal to the middle third of the maximal side of the rectangle.
Output: The two created rectangles.

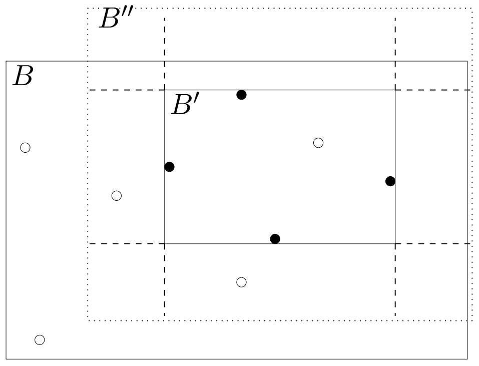
Remark 1 *Given as input a rectangle with constant aspect ratio, both processes output rectangles with constant aspect ratio. This remark will be useful in Section 4.*

See Fig. 1 for an illustration of the Sub-Rectangle process. The recursive method applies alternately the Sub-Rectangle and Cut-Rectangle processes to produce a decomposition of the original rectangle containing the input. The method stops when either the current rectangle contains one point or when the current rectangle contains no facilities, whichever happens earlier. We emphasize that in this section we do not use an apriori random shift of the coordinate system as Arora in [2]. The Sub-Rectangle process would diminish any randomization introduced at the beginning of the dissection. The randomization required by the upcoming Facts 1 and 2 is introduced by Cut-Rectangle. We also observe that the original rectangle is not necessarily covered by leaf rectangles in the decomposition, due to the sub-rectangle steps.

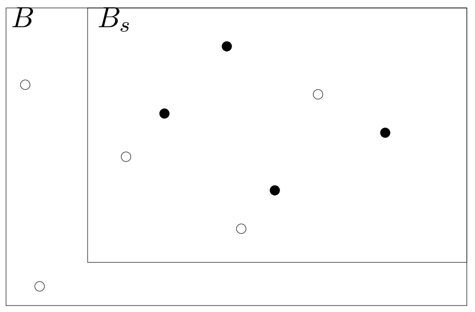
We place $m + 1$ evenly spaced points on each side of each rectangle in the dissection, where m will be defined later and depends on the accuracy ε of the sought approximation. We call these points *portals.* We define a *portal-respecting path* between two points to be a path between the two points that only crosses rectangles that enclose one of the points at portals. We define the *portal-respecting distance* between two points to be the length of the shortest portal-respecting path between the points. We begin by giving three technical lemmata which will be of use in the main Structure Theorem. Lemma 1 has a straightforward proof and gives the motivation behind the decomposition. We want short assignment edges in a given solution to be separated by rectangles of small sidelength.

Lemma 1 *If the first rectangle R in the dissection to separate points v and w has sidelength D, the difference between the portal respecting distance and the geometric distance between v and w is $O(D/m)$.*

We define a *cutting* line segment in the decomposition to be (i) either a line segment l that is used in the Cut-Rectangle process to divide a rectangle R into two rectangles or (ii) a line segment l used to form the boundary of a proper sub-rectangle R in the Sub-Rectangle procedure. In both cases we say that l *cuts* R. We define the *sidelength* of a cutting line l as the sidelength of the rectangle cut by l. Observe that the length of a cutting line is upperbounded by its sidelength.



(a)



(b)

Figure 1: Illustration of the Sub-Rectangle process. The circles are input points. The solid black ones indicate points with open facilities. (a) The intermediate rectangles created. (b) The input B and the output B_s of the process.

Lemma 2 *If any two parallel cutting line segments produced by the application of* Cut-Rectangle *are within distance L, one of the line segments has sidelength at most $3L$.*

Proof: Let l_1, l_2 be the two cutting segments at distance L. Assume without loss of generality that they are both vertical, l_1 is the longer of the two lines, l_1 is on the left of l_2 and cuts a rectangle R of sidelength greater than $3L$ into R_1 and R_2. Then l_1 is produced first in the decomposition. Thus l_2 is contained within R_2 (see Fig. 2), and since it comes second can only cut a rectangle R_2' contained within R_2. By the definition of Cut-Rectangle, if s is the sidelength of R_2', l_2 is drawn at least $s/3$ away from the left boundary of R_2' which implies $s/3 \leq L$. Thus $s < 3L$. □

The next lemma relates the length of a cutting line segment produced by Sub-Rectangle to the length of any assignment edges it intersects. We slightly abuse terminology and say that an edge (v, f) is *separated* when a cutting line separates v and f.

Lemma 3 *Let f be a facility and (v, f) an assignment edge of length D. If a cutting line segment σ produced by* Sub-Rectangle *separates v and f for the first time, then σ has sidelength at most $5D$.*

Proof: Let B_s be the rectangle cut by σ; let s be its sidelength. Assume without loss of generality that σ is a vertical line and that the horizontal dimension of B_s is maximal. Let B_s be produced by a Sub-Rectangle process with input B. We use the notation of the process for the intermediate

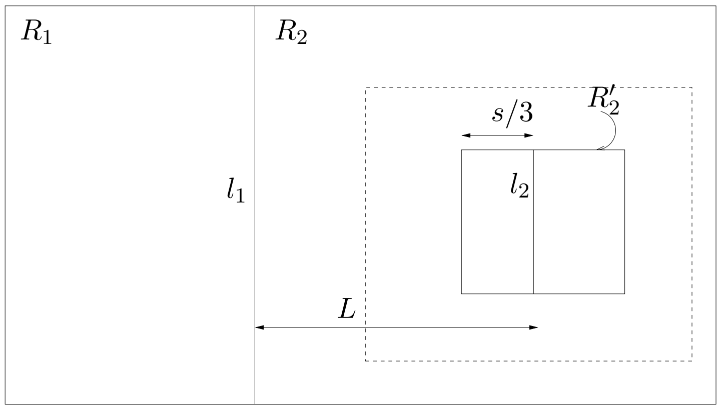


Figure 2: Illustration of the proof of Lemma 2. Rectangle R'_2 is a descendant but not necessarily a child of R_2 in the decomposition.

rectangles produced. Let $y \leq s$ be the sidelength of the rectangle B'. Then the sidelength of B'' is $(5/3)y > s$. The two vertical strips at the side of B'' are empty of facilities. Part of those strips may lie outside B, see Fig. 3. The intersection of the vertical strip which is crossed by (v, f) with B must be a rectangle whose projection on the x-axis has length $y/3$. Therefore $y/3 < D$ which implies that $s < 5D$. □

Consider the optimal solution with k given medians among the points. In the Structure Theorem we will prove the existence of a near-optimal solution in which a point v is always assigned to its closest or second-closest median. The *modified cost* of an assignment E is the sum over all the points of the portal-respecting distances to their respective assigned facility. Since the Structure Theorem assumes that the set of open facilities is given, it applies more generally to uncapacitated facility location.

We provide first two calculations that will be of use in the proof of the theorem.

Fact 1 *Let $\mathcal{E}(\Delta, Z)$ denote the event that an edge of length Δ is separated by a cutting line of sidelength Z that is produced by* Cut-Rectangle. *We have that*

$$Pr[\mathcal{E}(\Delta, Z)] \leq 3\Delta/Z.$$

Intuitively, sidelengths increase geometrically in the dissection. Therefore a consequence of Fact 1 is that the probability that an edge of length Δ is separated by a cutting line of sidelength Z or more that is produced by Cut-Rectangle is at most $O((\Delta/Z))$.

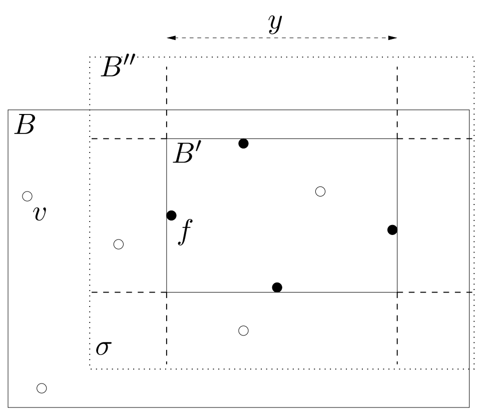
Fact 2 *Let $\mathcal{E}_{\geq}(\Delta, Z)$ denote the event that an edge of length Δ is separated by a cutting line of sidelength Z or more that is produced by* Cut-Rectangle. *Then*

$$Pr[\mathcal{E}_{\geq}(\Delta, Z)] = O(\Delta/Z).$$

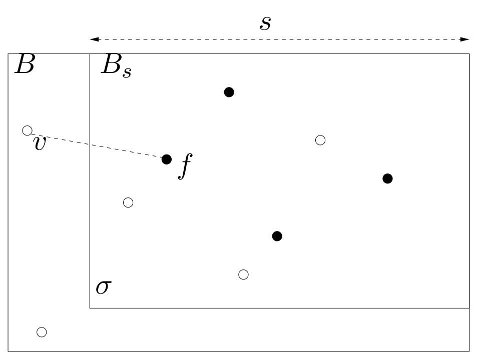
The following lemma will also be of use.

Lemma 4 *Let the distance from a point v to its closest open facility f be D and the distance from v to its second closest open facility l be L. In the decomposition, v and f are separated for the first time by a cutting line that has sidelength either (i) larger than $L/2$ or (ii) smaller than $8D$.*

Proof of lemma: We know that v and l are separated by the time the sidelength of any enclosing rectangle is at most L. Observe that by Lemma 3, a cutting line of sidelength $> 5D$ separating



(a)



(b)

Figure 3: Illustration of the proof of Lemma 3. The circles are input points. The solid black ones indicate points with open facilities. (a) The intermediate rectangles created by SUB-RECTANGLE. (b) The input B and the output B_s of the process.

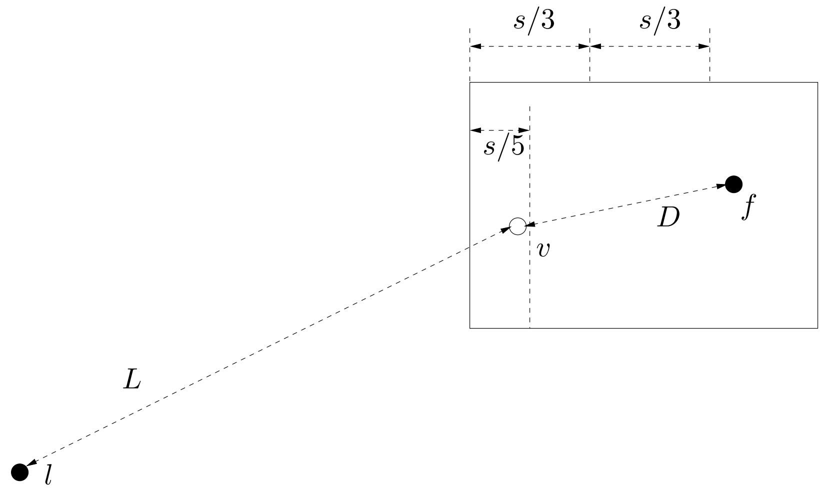


Figure 4: Illustration of the proof of Lemma 4. In the case we depict $s < 2D$.

v and f can only be produced by Cut-Rectangle. We will show that Cut-Rectangle cannot produce such a cutting line with sidelength in $(8D, L/2)$.

Without loss of generality we assume the relative positioning of v, f, l given above. See Fig. 4 for an illustration. For any rectangle of sidelength $L/2$ containing v, there are no facilities to the left of v. Thus, by the Sub-Rectangle process on input of sidelength $s < L/2$, the left boundary of any rectangle of sidelength $s < L/2$ that contains v is within distance at most $s/5$ of v. This is because in Sub-Rectangle the maximal rectangle empty of facilities is grown by at most a third in each direction. By the Cut-Rectangle process on input of sidelength s, any cutting line for a rectangle box is at least $s/3$ to the right of its left boundary which implies that the cutting line is at least $s/3 - s/5 = 2s/15$ to the right of v. Thus, the length D line segment cannot be cut until the sidelength of the enclosing rectangle is at most $(15/2)D$. □

Theorem 1 *(**Structure Theorem**) Let $m > 0$ be any integer and $F \subset N$ a set of open facilities. There is an assignment E such that the expected difference between the modified cost of E and the greedy cost C under F is $O(C \max\{1, \log(m)\}/m)$.*

Proof of theorem: By linearity of expectation, it suffices to bound the expected cost increase for a given assignment edge. For a point v we define $f \in F$ as v's closest facility and assume f to be to the right of v (without loss of generality.) We define $l \in F$ as the closest facility to the left of v. We denote the distance from v to f by D, and the distance from v to l by L. The idea behind the analysis of the portal-respecting solution is that assigning v to either f or l (a decision based on the amount by which the decomposition distorts each distance) will be enough to show near-optimal modified cost. The proof of the theorem shows how to actually construct the assignment E.

We assume without loss of generality that v and f are separated for the first time by a vertical cutting line. We can turn the configuration on its side and do the same argument if this condition does not hold.

The semicircle of diameter $2L$ centered at v and lying entirely to the left of the vertical line passing through v is empty in its interior. Therefore Lemma 4 applies.

We proceed to a case analysis based on which of the two edges (v, l) or (v, f) is separated first by the decomposition. Observe that (v, l) can be separated for the first time by either a vertical or a horizontal line. If v and f are separated for the first time by a line produced by Sub-Rectangle, we assign v to f in E and the increase in cost is $5D/m$ by Lemma 3. Therefore we can assume for

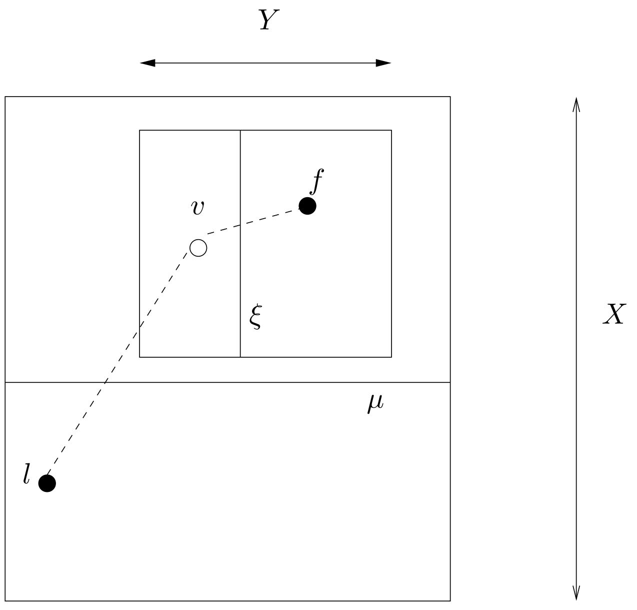


Figure 5: Case A2 in the proof of Theorem 1.

the remainder of the proof that v and f are first separated by a vertical cutting line produced by Cut-Rectangle. Let μ, ξ denote the lines separating for the first time (v, l) and (v, f) respectively.
CASE A. *Edge (v, l) is separated before (v, f).*

We will now calculate the expectation of the cost increase for the two possible subcases.
CASE A1. Edge (v, l) is separated for the first time by a vertical cutting line. We assign v to f in E. With some probability p, ξ has sidelength $L/2$ or more. By Lemma 4, ξ has sidelength at most $8D$ with probability $(1-p)$. Therefore, by Lemma 1, the cost increase is $O(D/m)$ with probability $(1-p)$. Now we turn to the case in which ξ has length $L/2$ or more. If μ is produced by Sub-Rectangle, by Lemma 3, μ has sidelength at most $5L$. If μ is produced by Cut-Rectangle, by Lemma 2 either ξ or μ has length at most $6L$. Moreover ξ has always length smaller than μ since it is produced second in the dissection. By Lemma 1 the cost increase is $O(L/m)$ regardless of the operation producing μ. By Fact 2, probability p is $O(D/L)$. Therefore the expected cost increase for CASE A1 is at most

$$(1-p)O(D/m) + pO(L/m) = O(D/m) + O((D/L)(L/m)) = O(D/m).$$

Remark 2 *The probability calculation for Case A1 depended only on the choice of which vertical line first cuts (v, f). This will be useful in Section 4 when we restrict our choices for vertical cutting lines.*

CASE A2. Edge (v, l) is separated for the first time by a horizontal cutting line. We assign v to f in E. We compute first the expectation of the cost increase conditioned upon the sidelength X of line μ. See Figure 5. By Fact 1, (v, f) is cut by a line of sidelength Y with probability at most $3D/Y$. Observe that this is true regardless of the value of X. Moreover $L/2 \leq Y \leq X$ or by Lemma 4, $Y \leq 8D$. The upper bound of X holds since (v, f) is contained in the rectangle cut by μ. For some constant c, the conditional expected cost increase is bounded by

$$\sum_{L/2 \leq Y \leq X | \exists i, Y = 2^i} (cD/Y)(Y/m) = O\left((D/m)\log(X/L)\right).$$

We now remove the conditioning on X. If line μ was produced by SUB-RECTANGLE, by Lemma 3 it has length at most $5L$. No matter how large the probability of this event is, by the sum above the cost increase is $O(D/m)$. If line μ is produced by CUT-RECTANGLE, by Fact 1 it has sidelength X with probability at most $3L/X$. The expectation of the cost increase is at most

$$\sum_{X \geq L | \exists i, X=2^i} 3(L/X)\log(X/L)O(D/m) = O(D/m).$$

Remark 3 *To compute the probability of the event B that ξ has a given sidelength Y, we only used the total probability theorem. We partitioned the event space into the possible events $A_1, A_2, \ldots$ based on the different possible values of X and then applied $Pr[B] = \sum_i Pr[B|A_i]Pr[A_i]$. This remark will be useful in Section 4.*

CASE B. *Edge (v, f) is separated before (v, l).*
CASE B1. Edge (v, l) is separated for the first time by a vertical cutting line. The difference from CASE A1 is that we cannot argue that ξ has smaller sidelength than μ therefore we follow a different strategy. If the sidelength of ξ is at most mL we assign v to f else we assign v to l in E.

By Lemma 4, if the cost increase exceeds $8D/m$ the sidelength of ξ is $L/2$ or higher. By Fact 1, there is a constant c such that assigning v to f yields an expected cost increase of

$$c[(2D/L)(L/2m) + (D/L)(L/m) + (D/2L)(2L/m) + \ldots + (D/mL)(mL/m)] = O(D\log(m)/m).$$

By Fact 2, the probability that ξ has a sidelength of mL or more and hence that we assign v to l is $O(\frac{D}{mL})$. In this case Lemma 2 gives that the sidelength of μ is at most $3(L + D)$. Therefore, when v is assigned to l, the expected assignment cost (and not just the increase) is

$$O\left(\frac{D}{mL}(L + L/m)\right) = O(D/m + D/m^2).$$

Therefore, regardless of whether v is assigned to f or l the expected cost increase is $O(D\log(m)/m)$.
CASE B2. Edge (v, l) is separated for the first time by a horizontal cutting line. We follow the same strategy as in Case B1: if the sidelength of ξ is at most mL we assign v to f else we assign v to l in E.

By the same argument as in Case B1, if v is assigned to f the expected cost increase is $O(D\log(m)/m)$. Consider the case where v is assigned to l. If μ is produced by SUB-RECTANGLE by Lemma 3 it has sidelength at most $5L$ and hence the expected assignment cost is $O(D/m+D/m^2)$ by a calculation similar to Case B1.

If μ is produced by CUT-RECTANGLE it cannot have sidelength less than L. Moreover by Fact 1 it has sidelength X with probability at most $3L/X$. X cannot exceed the sidelength Y of ξ but for every possible X in the range $[L, Y]$, the conditional probability that μ has sidelength X is still at most $3L/X$. Therefore we obtain that the expected assignment cost for v is

$$O\left(\sum_{mL \leq Y | \exists i, Y=2^i} \frac{D}{mL} \sum_{L \leq X \leq Y | \exists i, X=2^i} (L/X)(X + X/m)\right) = O\left(\frac{D}{mL}(L + L/m)\right) = O(D/m+D/m^2).$$

Remark 4 *To compute the probability of the event B that μ has a given sidelength X, we only used the total probability theorem. We partitioned the event space into the possible events $A_1, A_2, \ldots$ based on the different possible values of Y and then applied $Pr[B] = \sum_i Pr[B|A_i]Pr[A_i]$. This remark will be useful in Section 4.*

End of the proof of the Structure Theorem □

4 Modifying the Structure Theorem

The Structure Theorem in the previous section demonstrates that a portal-respecting $(1 + O(\log(m)/m))$-approximate solution exists while only placing m portals on the boundary of the decomposition rectangles. The randomization introduced by CUT-RECTANGLE is essential for the Structure Theorem. It does pose however the problem of how to enumerate all possible outputs of CUT-RECTANGLE in the context of a dynamic program as the one in [3]. In this section we show how to effectively bound the number of rectangles to be enumerated by the dynamic program and in the process obtain a nearly linear-time algorithm. To this end we introduce some discretization on the possible outcomes of SUB-RECTANGLE and CUT-RECTANGLE.

We give first some definitions. Consider the square of sidelength T that surrounds the original input. We know from Section 2 that $T = O(n^c)$, for some constant $c > 0$. We assume that $T = m2^\rho$ for some integer ρ and that the leftmost lower corner of the square lies at the origin of the axes. Call the vertical (horizontal) lines whose x-coordinate (y-coordinate) is an integral multiple of m *eligible.* Then, we call the vertical (horizontal) eligible lines with x-coordinate (y-coordinate) congruent to $0 \bmod 2^i$, $1 \leq i \leq \rho$, *i-allowable.* Note that the top and leftmost eligible lines are ρ-allowable, and that any j-allowable line is i-allowable for all $i < j$. A rectangle R of sidelength s is *t-allowable* if t is the maximum value such that: any two parallel sides of R of length s lie on t-allowable lines and $s \geq 2^t m$. Observe that when a rectangle is t-allowable, the aspect ratio is not bounded. The definition only guarantees that the smallest side has length $2^t m$. A rectangle which is t-allowable for some t and all whose sides have length within a factor of $5 + 1/2m$ of each other is called *allowable.*

We modify the SUB-RECTANGLE and CUT-RECTANGLE processes as follows.

SUB-RECTANGLE-NEW:
Input: An allowable rectangle containing at least one facility.
Process: Perform the SUB-RECTANGLE process of the previous section. Let B_s be the computed rectangle.
Output: the minimal allowable rectangle that contains B_s.

CUT-RECTANGLE-NEW:
Input: An allowable rectangle containing at least one facility.
Process: Choose a cutting line in the middle third of the maximal side of the rectangle, uniformly at random among all lines that produce two allowable sub-rectangles.
Output: The two allowable sub-rectangles.

To illuminate further CUT-RECTANGLE-NEW consider an example where it takes as input a t-allowable rectangle with sidelength $m2^t$. This implies that the rectangle is a square. Let us consider the horizontal side as maximal. The middle third of the maximal side has $\lfloor m/3 \rfloor$ candidate t-allowable cutting lines. Choosing one, yields two sub-rectangles for which the horizontal side has length at least $(m/3)2^t$ and at most $(2/3)m2^t$. Accordingly the two subrectangles are each i-allowable for $i \geq t - 2$.

Before proving the Modified Structure Theorem we examine the validity of the deterministic lemmata from Section 3 in the new setting. Clearly Lemma 2 continues to hold. We have to be more careful with Lemmas 3, 4 due to the extra requirement that the output of SUB-RECTANGLE-NEW should be allowable. This might cause the rectangle output by SUB-RECTANGLE to be "stretched". We show first that this stretch is small.

Lemma 5 *Each side of the rectangle output by* Sub-Rectangle-new *has length at most* $1 + 2/m$ *the length of the corresponding side of the rectangle* B_s *computed during the process.*

Proof: Within the Sub-Rectangle process, the rectangle B' of sidelength s is grown by $s/3$ in each dimension. It follows that the lengths of the sides of B_s are within a factor of $(s+2s/3)/(2s/3) = 5$ of each other. To meet the definition of an allowable rectangle, we only have to move the boundaries of B_s so that they fall on t-allowable lines. Here t is the maximum integer so that each side of B_s has length at least $2^t m$. To achieve this, we have to extend each side by at most 2^t in each direction for a total increase by a $1 + 2/m$ factor. $\square$

Lemma 4 continues to hold with the modified decomposition. In fact it becomes slightly stronger since (cf. Proof of Lemma 4) the left boundary of any allowable rectangle produced by Sub-Rectangle-new that has sidelength $s < L/2$ and contains v is within distance at most $s(1/5 + 1/m)$ from v. We now give the equivalent to Lemma 3 in the modified setting.

Lemma 6 *Let f be a facility and (v, f) an assignment edge of length D. If a cutting line segment σ produced by* Sub-Rectangle-new *separates v and f for the first time, then σ has sidelength at most* $(5 + 6/m)D$.

Proof: We outline only the changes from the Proof of Lemma 3. Keeping the same notation, by Lemma 5 $(5/3)y \leq s \leq (5/3 + 2/m)y$. Since $y/3 < D$, we obtain $s < (5 + 6/m)D$. $\square$

The difference between Lemma 3 and 6 is negligible from the point of view of the Modified Structure Theorem since it only introduces an additive $O(D/m^2)$ error term. We can now focus on the probabilistic part of the modified decomposition.

The new dissection is similar to the one from Section 3. The primary difference is that the randomization in the Cut-Rectangle process has been diminished. If we add back some randomization up front by shifting the original rectangle containing the input, we can get the same result as in the Structure Theorem on the expected increased cost of a portal respecting solution. We define an (a, b)-*shifted* coordinate system, $0 \leq a, b \leq T$, one in which the x and y coordinates are shifted by a and b respectively. The shifting uses wraparound as in [2]: a vertical line which had x-coordinate x_1 before the shifting has coordinate $x_1 + a \bmod T$ after. If a and b are chosen at random, any vertical or horizontal line is equally likely to be i-allowable. The *new modified cost* of an assignment is defined with respect to the new dissection give above.

Theorem 2 *(***Modified Structure Theorem***) Let $m > 0$ be any integer and $F \subset N$ a set of open facilities. If the coordinate system is randomly shifted by (a, b) where a and b are chosen independently in $[0, T]$ then there is an assignment E such that the expected difference between the new modified cost of E and the greedy cost C under F is* $O(C \max\{1, \log(m)\}/m)$.

Proof: As mentioned, minor variations of the deterministic lemmata from Section 3 continue to hold in the restricted version of the dissection. The randomized portion in the proof of Theorem 1 only reasons about two types of events. Moreover, it reasons about each event in isolation (cf. Cases A1 and A2 in the proof, the rest are similar). Thus, we need only be concerned with the probabilities of each event. The random shift up front of the coordinate system along with the randomization inside the process will ensure that these two types of events occur in our decomposition into allowable rectangles with approximately the same probability as in the previous decomposition.

The first event (cf. Case A1) is that a line produced by Cut-Rectangle-new of length X cuts a line segment of length D. The probability of this event is required to be at most $3D/X$ in the proof of the Structure Theorem. We show that this continues to hold albeit with a constant larger than 3.

We assume for simplicity, that the line segment is in a rectangle R of sidelength exactly $X = m2^i$ at some point. (This will be true to within a constant factor.) If $D < 2^i$, we know that the segment is cut by at most one i-allowable line. The probability of this event is at most $D/2^i$ due to the random shift. The CUT-RECTANGLE-NEW process chooses from $m/3$ i-allowable lines uniformly at random. Thus, the line segment is cut with probability $1/(m/3)$ times $D/2^i$, which is $3D/X$ as required. If $D > 2^i$, we notice that D intersects at most $\lceil D/2^i \rceil < 2D/2^i$ i-allowable lines. By the union bound the probability that this line segment is cut during CUT-RECTANGLE-NEW on R is upperbounded by $2D/2^i$ times $3/m$, i.e., $6D/X$.

The second event (cf. Case A2) is the intersection of two events; a horizontal line of length X cuts a segment of length L and a vertical line of length Y cuts a segment of length D. In the proof of the Structure Theorem, the probability was shown by the total probability theorem to be upper bounded by the product of the probability bounds of the two events, i.e., $3L/X$ times $3D/Y$. The second term represented the conditional probability that edge (v, f) is cut by a line of sidelength Y given that (v, l) was cut by a horizontal line of sidelength X. We argued that the conditional probability does not depend on X.

For our restricted decomposition, the probability of each event in isolation can be bounded by $6L/X$ and $6D/Y$ as argued above. Moreover, we chose the horizontal and vertical shifts independently and we choose the horizontal and vertical cut-lines in different processes. Thus, we can also argue that the probability of the intersection of the two events is at most $6L/X$ times $6D/Y$. $\Box$

We now prove a lemma bounding the number of allowable rectangles.

Lemma 7 *The number of allowable rectangles that contain l or more points from the input set N is $O(m^4(n/l)\log n)$.*

Proof: Our proof uses a charging argument. Let R_l be a rectangle on the plane that has minimum sidelength, say L and contains l points. We bound the cardinality of the set S_l of allowable rectangles, which are distinct from R_l, contain at least l points and have at least one point in common with R_l. Let R_a be such a rectangle. Then R_a has sidelength at least L, otherwise it would have been chosen instead of R_l.

We bound the number of allowable rectangles in S_l with sidelength $X \in [2^{i-1}, 2^i]m$ by $O(m^4)$ as follows. The corners must fall on the intersection of two t-allowable lines, that are within distance X from some side of R_l. Since allowable rectangles have bounded aspect ratio, the possible values for t are 2^{i-k}, $k = 0, 1, 2, 3$.

The number of j-allowable lines that are within distance X from some side of R_l is $O(X/2^{j-1})$ since $X \geq L$. Thus, the number of corner choices is $O(m^2)$. Two corners must be chosen, so the number of rectangles in S_l of sidelength $X \in [2^{i-1}, 2^i]m$ is $O(m^4)$. Since there are $O(\log n)$ values of i, $|S_l| = O(m^4 \log n)$.

Now, we remove R_l and its points from the decomposition, and repeat the argument on the remaining $n - l$ points. The number of repetitions until no points are left is $O(n/l)$ therefore by induction, we get a bound of $O(m^4(n/l)\log n)$ on the number of allowable rectangles that contain at least l points. $\Box$

5 The dynamic program

We have structural theorems relative to a particular decomposition. Unfortunately, the decompositions are defined with respect to the facility locations. However, in reality they only use the facility locations in the SUB-RECTANGLE steps. Moreover, the number of sub-rectangles is at most polynomial in the size of the original rectangle. Indeed, the number of allowable sub-rectangles is polynomial in m and n. Thus, we can perform dynamic programming to find the optimal solution. The structure of the lookup table is similar to the one used in [3]. We exploit our Structure Theorem and the analysis on the total number of allowable rectangles to obtain a smaller number of entries.

We will develop two versions of our dynamic program, in order of increasing sophistication, each with a somewhat different time complexity. Our dynamic programs have much in common with the one in [3]. For the sake of completeness we include here all the relevant definitions from [3].

5.1 The table definition

In all our approaches the table will consist of a set of entries for each allowable rectangle that contains at least one point. There is a one-to-one correspondence between entries of the table and subproblems on the corresponding rectangle. Subproblems on a given rectangle are generated by what amounts to a guessing (enumeration) of the location of facilities that serve the points in the rectangle. These facilities could be either inside the rectangle or outside. In either case, since by the Structure Theorem it suffices to consider portal-respecting solutions, we only store the *interaction pattern* (defined by Items 2 and 3 below) of the portals to the facilities. For each allowable rectangle, we will enumerate the following:

1. the number of facilities in the rectangle,
2. a distance for each portal p to the nearest facility in the rectangle and
3. a distance for each portal p to the nearest facility outside the rectangle, if one exists that is nearer to p than all facilities inside the rectangle.

Every triple of this sort defines a subproblem. Given a subproblem which is indexed by f for the number of facilities and is defined on a rectangle R, a solution will be: a set of f facilities among the points in R and a portal-respecting path from each point in R to one of the facilities in R or to a portal on the boundary. The table entry corresponding to the subproblem will store the minimum service cost under the given constraints. Recall from Section 3 that the decomposition stops as soon as a rectangle is produced that contains no facilities. Formally, a *leaf* of the decomposition is a rectangle R which has been produced by the decomposition, such that either (i) R contains one point and this point is an open facility or (ii) R is a rectangle that contains at least one point and no facilities.

The enumeration of the distances happens by using the *inside* and *closest* functions on the portals as defined in [3]. In order to save on the size of the table the definitions of the two functions follow three rules, same as in [3]. We present them first, establish their validity and then provide the definitions that implement them.

Rule 1: we will only approximate the distances to the nearest facility inside and outside of a rectangle R to a precision of s/m for a rectangle of sidelength s.

Rule 2: as far as the distance from a portal p to the closest facility is concerned we do not consider distances of more than $6s$. Clearly the distance from a portal p on the boundary of R to the nearest facility within R cannot exceed $2s$. Therefore this rule affects only assignments to the outside of R, i.e., the upcoming definition of the *closest* function. We want to ensure that no assignments are lost for the dynamic program. A key remark is that a portal p is assigned to a facility η outside R *only if all* facilities within R (if such exist) are further from p than η. Therefore if R contains some facility, any facility outside R at distance more than $2s$ is of no interest to the points in R. If R contains no facilities, it is a leaf of the decomposition. This leaf rectangle must have been produced

by Cut-Rectangle-new. Let R' be the parent rectangle R' that was cut. $R' - R$ must contain some facility, else R' itself must have been a leaf. The sidelength s of R is at least one third the sidelength of R'. Hence by a generous estimate there is a facility at distance at most $6s$ from every portal of R.
Rule 3: it suffices for neighboring portals to represent distances as offsets from the previous distance. This is ok because the distance value at a portal only changes by a constant from the distance value at an adjacent portal.

We are now ready to give the definitions of the the *inside* and *closest* functions as in [3]. Let Π be a subproblem defined on rectangle R and let Π be indexed by a number of facilities $f \in [0, k]$. For a portal p, *inside* (*closest*) encodes the distance to the nearest facility to p within (outside) R, subject to Rules 1, 2, and 3.

- If $f = 0$, we set $\mathit{inside}(p) = \infty$ for every portal p on the boundary of R. This should happen for all R that are leaves of the decomposition.
- If $f > 0$ we define an assignment *inside* of numbers $[1, \ldots, 4m]$ to the portals of R, such that $|\mathit{inside}(p) - \mathit{inside}(p')| \leq 1$ if portals p and p' are successive along the boundary of R. The domain of the assignment expresses Rule 1, i.e., the precision of the distances. The condition on the difference expresses Rule 3, i.e., we represent distances as offsets from the previous distance.
- If $f < k$ we define an assignment *closest* of numbers $[1, \ldots, 4m]$ to the portals of R, such that (i) $|\mathit{closest}(p) - \mathit{closest}(p')| \leq 1$ if portals p and p' are successive along the boundary of R and (ii) $\mathit{closest}(p) < \mathit{inside}(p)$. Since by Rule 2 above, we can always assume that there is a facility within distance $6s$ from p, the definition can only fail if one such facility within R is the nearest to p, i.e. when Condition (ii) fails. In this case we set $\mathit{closest}(p) = \infty$.
- If $f = k$, we set $\mathit{closest}(p) = \infty$ for every portal p on the boundary of R.

By the three rules above, as far as the distances are concerned we only need to guess $O(m)$ bits per entry. Taking into account the guess for the number of facilities the total number of table entries (subproblems) corresponding to each allowable rectangle is bounded by $k2^{O(m)}$. We bound the table size by noting that the total number of allowable rectangles is, by Lemma 7, at most $O(m^4 n \log n)$. Thus, we can bound the total number of entries in the table by $k2^{O(m)} n \log n$.

5.2 Computing the table entries

With respect to the recursive decomposition defined by the Modified Structure Theorem every subproblem defined on a rectangle R has one or two *children* subproblems, the number depending on which process, Cut-Rectangle-new or Sub-Rectangle-new, is applied. The solution for a subproblem Π defined on rectangle R is computed by looking at children subproblems whose interaction patterns match with the interaction pattern of Π. The *recursion height* of any subproblem defined on R is equal to the maximum length of a path leading to a leaf of the decomposition in the tree representing the dissection defined by the Modified Structure Theorem. The maximum recursion height is obviously $O(\log n)$. Based on which process is applied, we are looking for either (i) a pair of subproblems defined on two allowable subrectangles that cover R and whose total number of facilities equals that of Π or (ii) a subproblem defined on a single allowable subrectangle of R whose number of facilities equals that of R. Among the matching combinations we select the one of minimum cost, where the cost is computed as follows. After fixing the values of the *inside* and *closest* functions on the portals of Π, for every point in R, we know the location of the nearest facility, either within R or outside. For a portal p on the boundary of R let $n(p)$ be the total number of points assigned to p, i.e., the total number of points that are served by a facility on the outside via p. Similarly to [3], the *cost* of Π equals the total assignment cost within R plus

$$\sum_{\text{portals } p} closest(p)n(p)(s/m).$$

An important difference of our scheme from [3] is that we cannot apriori compute the randomly shifted dissection and then apply dynamic programming on the rectangles of the dissection. Because of the Sub-Rectangle-new process, for any given subproblem we have to enumerate all possible outputs of either the Sub-Rectangle-new process or all different cuttings of the Cut-Rectangle-new process. This gives rise to two different types of algorithmic steps, one for each case. Fortunately, by Lemma 7 the total number of rectangles to be enumerated throughout the dynamic program is $O(m^4 n \log n)$.

Let R be a rectangle, of sidelength s, at height i of the recursion and assume inductively that the algorithm has solved all subproblems at height $i-1$. Consider a subproblem Π defined on R and let Π be indexed by a number f of facilities. We now give the details of the algorithm's steps.

Application of Sub-Rectangle-new. To capture the outcome of Sub-Rectangle-new the algorithm enumerates all possible subproblems on allowable sub-rectangles R_u of R such that

S1. the number of facilities open in R and R_u is equal to f.

S2. for each portal p of R there is a portal p' of R_u such that $d(p,p') + inside(p')(s_u/m) \leq inside(p)(s/m)$. Here s_u denotes the sidelength of R_u, with $s_u \leq s$.

S3. for each portal p of R_u there is a portal p' of R such that $d(p,p') + (s/m)closest(p') \leq closest(p)(s_u/m)$.

There does not seem to be a way to lookup the cost of R, say as a function of the cost of the chosen R_u. Moreover the number of possible candidates for R_u could be large; we need to consider candidates of arbitrarily small sidelength. We take the inverse view and enumerate the work done for each allowable rectangle S when it is considered as a subrectangle of some R. The argument that follows is due to J. Remy.

For fixed t, any rectangle S is contained in $O(1)$ t-allowable rectangles of constant aspect ratio. Therefore a fixed S is considered as a subrectangle $O(\log n)$ times. Therefore by Lemma 7 the total number of subrectangles, not necessarily distinct, that the dynamic program needs to enumerate is $O(m^4 n \log^2 n)$. For every such occurrence of S in the dynamic program we enumerate all the interaction patterns and the possible number of facilities which yields $k2^{O(m)}$ subproblems. We proceed to bound the running time for each subproblem involving a rectangle R containing S.

A subproblem defined on S specifies the positions of facilities within R (up to an additive $sidelength(S)/m$ factor). We want to assign the points in R to the closest facility within S or outside R without paying time proportional to the number of points in R. The intuition behind the proof is that a higher additive error on the assignment can be tolerated for points in $R \setminus S$ that are "far" from S. The adaptive dissection suggests at a high level to guess the rectangle S and accordingly the location of facilities within R. To assign the points efficiently, the algorithm will be also adaptive at a much lower level and tune the accuracy of the assignment for the different points.

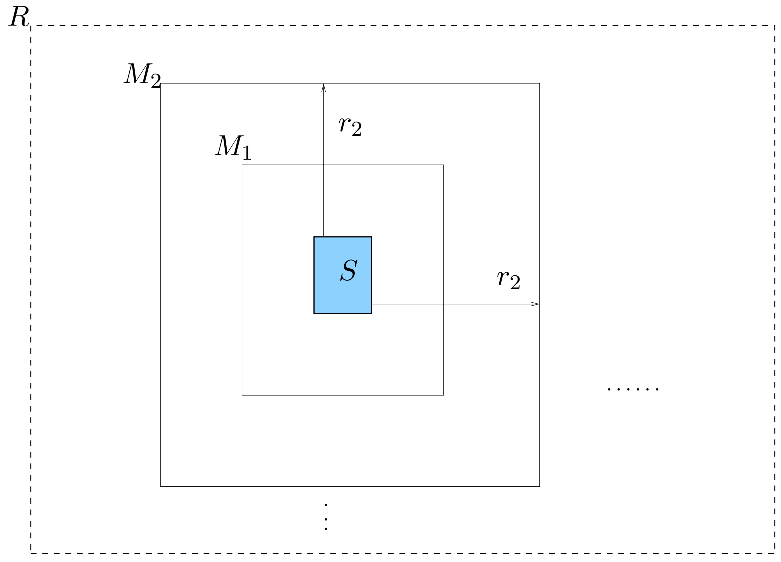


Figure 6: Illustration of the moat construction process. If all sides of S are equidistant from the sides of R, each of the moats will be topologically equivalent to an annulus.

Lemma 8 (due to J. Remy) *Let Π_R and Π_S be two subproblems defined on rectangles R and S where S is contained in R. The two subproblems agree as on S1–S3 above. Let G be the assignment of every point in R to its closest facility where the locations of facilities are specified by Π_R and Π_S. We can compute an assignment G' such that the service cost of each point p has an additive error of $O(1/m)$ times the modified service cost of p in G. G' and its total cost can be computed in $O(m^4 \log^2 n)$ time worst-case.*

Proof: The construction uses some ideas similar to the ones in [26]. We partition the area of $R \setminus S$ into rectangular *moats* $M_1, M_2, \ldots$. Moat M_1 is the $2d$-shape created by surrounding S by a rectangle S' that strictly contains S and then removing S. Moat M_{i+1} is the area remaining from a rectangle surrounding moat M_i after $(\bigcup_{j\leq i} M_j) \cup S$ has been removed. The *radius* r_i of moat M_i is the maximum distance of an outer vertical (or horizontal) side from the nearest vertical (horizontal) side of S. The radii are set to be geometrically increasing with $r_i = \mathit{sidelength}(S)(1 + 1/m)^i$, $i = 1, 2, \ldots$. Any outer vertical (horizontal) side of M_i is at the maximum distance that does not exceed r_i from the closest vertical (horizontal) side of S. Hence a side at a distance less than r_i will fall on the boundary of R. See Fig. 6 for an illustration. If the area left uncovered in R at some point of the process is not wide enough to become a moat by itself, we append it to the last moat produced and terminate. The number of moats is $O\left(m \log(\mathit{sidelength}(R) - \mathit{sidelength}(S))\right) = O(m \log n)$.

Each moat M_i is divided into $O(m^2)$ rectangular cells whose sides are each $\Theta(r_i/m)$ long. See Fig. 7. The total number of cells across all moats is $O(m^3 \log n)$. For every point in $R \setminus S$ we have to try $O(m)$ portals on the boundaries of S and R to find the closest facility. The algorithm identifies *all points within a cell C* with the center of the cell and assigns them to the same portal. We calculate now the error induced by this collapsing of points. For a point p in cell C of moat M_l, let d_1 be its distance from the nearest portal on the boundary of S. The estimated distance d_1' for all points in the cell to the closest portal on the boundary of S will be at most

$$d_1 + O(\mathit{sidelength}(C)) \leq d_1 + O\left(\frac{\max\{(1+1/m)d_1, (1+1/m)\mathit{sidelength}(S)\}}{m}\right) = d_1 + O(1/m)\max\{d_1, \mathit{sidelength}(S)\}.$$

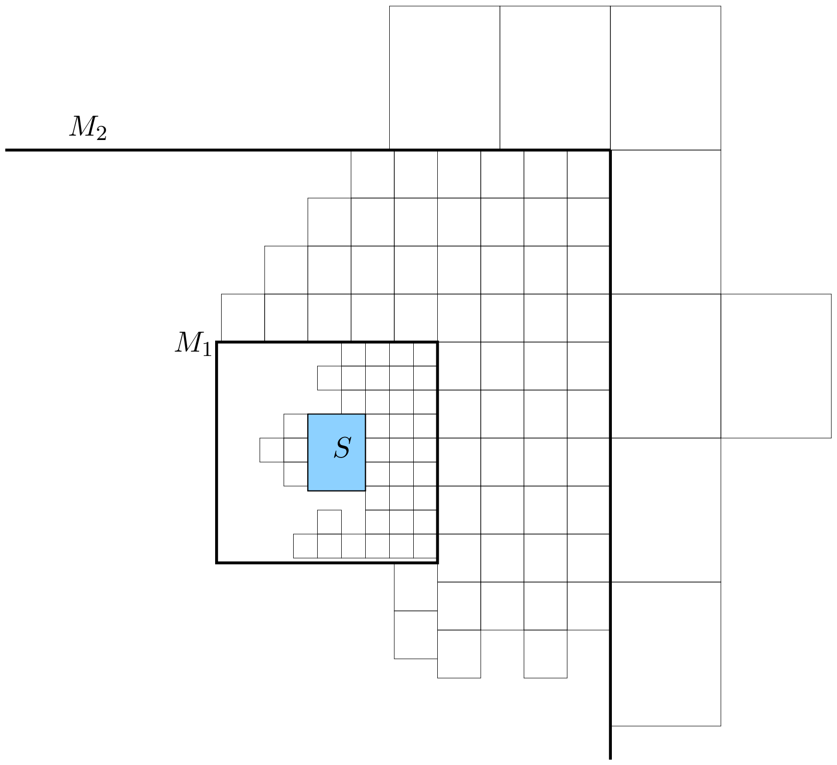


Figure 7: Illustration of the cell construction process.

The second term in the max expression above is needed for points in M_1. For the same point p let d_2 be its distance from the nearest portal on the boundary of R. In the worst-case the distortion is maximized when M_l is the outer moat, i.e., the one touching on the boundary of R. The estimated distance after collapsing p will be at most $d_2 + O(sidelength(C)) \leq d_2 + O(sidelength(R)/m)$ which is fine since the modified service cost through a portal on the boundary of R is already distorted by $O(sidelength(R)/m)$.

The total cost of assigning the points in $R \setminus S$ is the weighted sum of the assignment costs of the cells where the cost for each cell is weighted by the number of points in the cell. Computing this number can be done by orthogonal range searching techniques in $O(\log n)$ time per cell (see [1]) after an $O(n \log n)$ global preprocessing. Therefore the total running time is $O(m^4 \log^2 n)$. □

Remark 5 *The analogue of Lemma 8 holds in d-dimensional space. The number of cells is $O(M^{d+1} \log n)$ where M is the number of portals maintained on each face of a parallelepiped. The range searching query time increases by a multiplicative $O(\log^{d-2} n)$ factor [1]. The running time becomes $O(M^{d+2} \log^d n)$.*

By Lemma 8 and the previous discussion the total computation time of the dynamic program due to Sub-Rectangle-new is

$$\begin{aligned}\text{Total time due to Sub-Rectangle-new} &= \quad (\text{Time per entry}) \times (\text{number of entries}) \qquad (2)\\ &= O(m^4 \log^2 n) \times k2^{O(m)} n \log^2 n = O(k2^{O(m)} n \log^4 n).\end{aligned}$$

This bound can be improved by amortization. By Lemma 7, the number of allowable rectangles that contain k or more points is $O(m^4 (n/k) \log n)$. If an allowable rectangle S contains fewer than

$l < k$ points from N, we only need to keep $l2^{O(m)}$ table entries for it, since at most l facilities can be placed inside it. Moreover, the number of allowable rectangles containing between l and $2l$ points is shown by Lemma 7 to be $O(m^4(n/l)\log n)$. We can now bound the total time due to SUB-RECTANGLE-NEW by

$$O(m^4 \log^2 n) \times \left(k2^{O(m)}(n/k)\log^2 n + \sum_{l=2^i}^{l<k} l2^{O(m)}(n/l)\log^2 n\right) = 2^{O(m)} n \log^4 n \log k. \quad (3)$$

Application of Cut-Rectangle-new. The algorithm enumerates also all possible cuttings of R according to the CUT-RECTANGLE-NEW process. It looks for combinations of subrectangles R_1, R_2 of R whose interaction patterns match with that of Π. The details are similar with the method outlined in [3], where four children rectangles are considered instead of two. In particular the algorithm enumerates pairs of subproblems defined on allowable rectangles R_1, R_2 that partition R, fulfill the sidelength requirements of CUT-RECTANGLE-NEW and in addition fulfill the following:

C1. the sum of the number of facilities in R_1 and R_2 equals f.

C2. for each portal p of R there is a portal p' on one of R_1, R_2, call it R', such that $d(p,p') + \mathit{inside}(p')(s'/m) \leq \mathit{inside}(p)(s/m)$. Here s' is the sidelength of R', with $s' \geq s/3$.

C3. for each portal p of R_j, $j = 1, 2$, there is either (i) a portal p' on one of R_1, R_2, call it R', such that $d(p,p') + \mathit{inside}(p')(s'/m) \leq \mathit{closest}(p)(s(j)/m)$ or (ii) a portal p' of R such that $d(p,p') + \mathit{closest}(p')(s/m) \leq \mathit{closest}(p)(s(j)/m)$. Here $s(j)$ is the sidelength of R_j.

The total number of combinations of subrectangles and interaction patterns to consider for Π, due to CUT-RECTANGLE-NEW is $2^{O(m)}$. We also need to search over all possible splits of the facilities across pairs of subrectangles which gives an additional k factor. Therefore the total number of subproblems needed to determine the value for the entry of subproblem Π is $k2^{O(m)}$. In contrast to the SUB-RECTANGLE-NEW case, computing the cost of every combination can be done via lookup. For every pair of children subproblems that fulfill C1-C3, the cost of Π is equal to the sum of the two costs. We have that

$$\begin{aligned}\text{Total time due to CUT-RECTANGLE-NEW} &= (\text{Time per entry}) \times (\text{number of entries}) \quad (4)\\ &= k2^{O(m)} \times k2^{O(m)} n \log n = O(k^2 2^{O(m)} n \log n).\end{aligned}$$

Again we can improve the total time to $O(k2^{O(m)} n \log k \log n)$ by amortization. By Lemma 7, the number of allowable rectangles that contain k or more points is $O(m^4(n/k)\log n)$. If an allowable rectangle contains fewer than $l < k$ points from N, we only need to keep $l2^{O(m)}$ entries for it, since at most l facilities can be placed inside it. Moreover, the number of allowable rectangles containing between l and $2l$ points is shown by Lemma 7 to be $O(m^4(n/l)\log n)$. We can now bound the total number of table entries by

$$k2^{O(m)} O(m^4(n/k)\log n) + \sum_{l=2^i}^{l<k} 2^{O(m)} O(m^4(n/l)\log n) l = O(2^{O(m)} n \log k \log n). \quad (5)$$

By Equations (4), (5) the total running time due to CUT-RECTANGLE-NEW is

$$O(k2^{O(m)} n \log^2 n). \quad (6)$$

By (3) and (6) the total running time of the algorithm is $O(\max\{k, \log^3 n\} 2^{O(m)} n \log^2 n)$ worst-case.

5.3 Improving the dependence on k

In a second approach, we can further replace the factor of k with a factor that depends only on m and $\log n$. The idea is to use indexing according to approximate cost. In this approach, the table entries are indexed by

1. the total cost of the solution corresponding to the service cost required by the clients (points) of the rectangle. This cost includes the possible assignments of points to facilities outside the rectangle via the appropriate portals.

2. a distance for each portal p to the nearest facility in the rectangle and

3. a distance for each portal p to the nearest facility outside the rectangle, if one exists that is nearer to p than all facilities inside the rectangle.

The value of a table entry is the minimum number of facilities that are required inside the rectangle to find a solution with the corresponding cost and interaction pattern. Naturally, we disregard subproblems indexed by cost values that are too small to be attainable with no more than k facilities.

How many different cost indexes are there? Given our initial assumption about the points lying on a grid of polynomial size, the total number of interesting cost values is polynomial in n. We quantize them further by considering only cost values that are powers of $(1 + \delta)$. Then the number of distinct cost values becomes $\Theta(\log_{1+\delta} n)$. The effect of this quantizing is that for a single subproblem we only approximate its cost value to within a factor of $(1 + \delta)$. These approximations pile up multiplicatively over the $O(\log n)$ levels of the decomposition. To offset this effect we choose δ to be $\Theta(m^{-1}/\log n)$. The total number of cost indexes is then $\Theta(m \log^2 n)$. Accordingly, the total number of table entries is $(\log^2 n)2^{O(m)} n \log n = 2^{O(m)} n \log^3 n$.

Under the cost quantizing, the intended meaning of the table values is the following. Let an entry be indexed by cost $C = (1 + \delta)^y$ and some interaction pattern. Let this entry correspond to height i of the recursion. The value f in the entry means that the minimum number of facilities to achieve cost at most $C/(1 + \delta)^i$ is at least $f + 1$. In Lemma 9 below we will show that this intention can indeed be achieved. Computing an entry involves searching over combinations of subrectangles, matching interaction patterns, and cost splits, and choosing the combination that requires the minimum number of facilities in order to be realized. Same as before, the number of matching interaction patterns and subrectangles to be considered is $2^{O(m)}$ per table entry. The dynamic program computes entries defined on the same rectangle and with the same interaction pattern by increasing order of cost index.

For the number of cost splits we reason as follows. Given a subproblem on a rectangle R indexed by interaction pattern P and cost C one needs to consider all three cases that follow. *Cost-Split Case (i):* all subproblems on R indexed by the same interaction pattern P and costs less than C. *Cost-Split Case (ii):* due to the Sub-Rectangle-new process, all subproblems defined on subrectangles of R with matching interaction pattern and costs less than or equal to C. Note that we might consider also cost equal to C if the subrectangle might contain all the points of R. *Cost-Split Case (iii):* due to the Cut-Rectangle-new, all cost splits C_1, C_2, across a pair of rectangles with appropriate interaction patterns, such that $C_1 + C_2 = C$. The minimum number of facilities between (i), (ii) and (iii) will give the value for the entry. Because everything is quantized at powers of $(1 + \delta)$ sums get distorted. Therefore for (iii) instead of considering all cost splits such that $C_1 + C_2 = C$ we consider C_1, C_2 such that

$$C/(1 + \delta) < C_1 + C_2 \leq C.$$

The number of such splits is proportional to the total number of distinct cost values, i.e., $O(m \log^2 n)$.

To obtain the new running times we make the necessary changes to Equations 2 and 4. In Equation 4 the k factor is replaced throughout with $O(m \log^2 n)$, yielding time $O(2^{O(m)} n \log^5 n)$. In Equation 2 the corresponding number of entries is proportional to the square of the total number of cost values. Therefore the total time due to Sub-Rectangle-new is $O(2^{O(m)} n \log^8 n)$. Taking the maximum over the Sub-Rectangle-new and the Cut-Rectangle-new processes yields a worst-case running time of $O(2^{O(m)} n \log^8 n)$.

Now, we should argue that at each level the table entry corresponds to the minimum number of facilities that achieve the specified connection cost or better. This follows easily from the following fact.

Lemma 9 *If the table entry for a subproblem Π, which is indexed by cost C is filled with the number f, then there exists a set of f facilities within the corresponding rectangle and a portal-respecting assignment of points to facilities that satisfies the interaction patterns such that the total cost is no more than C. Furthermore, there is no portal-respecting solution for Π that (i) uses fewer facilities and (ii) has actual service cost less than or equal to $C/(1+\delta)^i$ where i is the recursion height of the subproblem Π.*

Proof: We use induction on i. It is easy to compute table entries for rectangles that are leaves of the decomposition. Then, we inductively assume the lemma, and note that the first part holds by construction. For the second part we consider the way the dynamic program computed the entry for Π. Let R be the corresponding rectangle. For the sake of contradiction assume that there is a better solution for Π, i.e., one with $g < f$ facilities and cost $C^* \leq C/(1+\delta)^i$. Among these solutions, choose the one that minimizes g. By our (Modified) Structure Theorem it must be decomposable either via Sub-Rectangle-new into a subproblem Π_u or via Cut-Rectangle-new into two subproblems Π_1, Π_2.

In the first case, let R_u be the subrectangle of R on which Π_u is defined and let $C_u \leq C$ be the minimum cost for which Π_u admits a solution with g facilities. The cost C^* equals C_u plus the assignment cost of the points in $R - R_u$. By induction there is a computed entry for Π_u filled with the value g and indexed by cost $C'_u \leq (1+\delta)^{i-1} C_u \leq (1+\delta)^{i-1} C^* \leq C/(1+\delta)$. Π_u will be examined by the dynamic program as per Cost-Split Case (ii) above. This contradicts the fact that the dynamic program missed the solution with the g facilities.

In the second case, this better solution is defined by combining the solutions to the two subproblems Π_1 and Π_2. Then, by induction we have solutions with the correct number of facilities in the entries for Π_1 and Π_2 which are indexed by the appropriate approximate cost, i.e., the table entries are indexed by costs C_1 and C_2 that are at most $(1+\delta)^{i-1}$ times the actual assignment cost for each subproblem. We show that at combination one can further lose an additional factor of at most $(1+\delta)$. Let Γ be the smallest power of $(1+\delta)$ such that $C_1 + C_2 \leq \Gamma$. The idea is that this combination of Π_1 and Π_2 with actual service cost at most $C_1 + C_2$ will be considered by *any* subproblem, with appropriate interaction pattern, defined on R at recursion height i and whose cost index is at least Γ. In particular we have that

$$\Gamma \leq (1+\delta)(C_1 + C_2) \leq (1+\delta)^i C^* \leq C.$$

If $\Gamma < C$, (recall C is also a power of $(1+\delta)$), the induction step gives us that the combination was examined as per Cost-Split Case (iii) when filling out the entry of the subproblem on R, with the same interaction pattern as Π but indexed by cost Γ. Therefore it will be available when filling out the entry for Π and will be examined as per Cost-Split Case (i), a contradiction.

If $\Gamma = C$, we have that $C/(1+\delta) < C_1 + C_2 \leq C$. Therefore the combination will be examined when filling out the entry for Π as per Cost-Split Case (iii). Again this contradicts the fact that the dynamic program missed this solution. □

We are now ready to state the main result of the paper.

Theorem 3 *Given an instance of the k-median problem in the 2-dimensional Euclidean space, and any fixed $m > 0$, there are randomized algorithms that compute a $(1+O(\log(m)/m))$-approximation, in expectation, with worst-case running times $O(2^{O(m)} n \log^8 n)$ and $O(2^{O(m)} \max\{k, \log^3 n\} n \log^2 n)$ respectively.*

Repeating the algorithm $O(m \log n)$ times gives a $(1 + O(\log(m)/m))$ approximation guarantee with probability $1 - o(1)$. The algorithm can be easily extended to instances in the d-dimensional Euclidean space. The main difference is that now we work with rectangular parallelepipeds whose faces lie on i-allowable hyperplanes. As a consequence the bound given in Lemma 7 increases by an $O(m^{2d-4})$ factor and in the dynamic program we have to maintain data for each of the $2d$ faces of each parallelepiped where each face contains m^{d-1} portals. Moreover, the increased cost for range searching in Lemma 8 has to be taken into account. Following the same steps as in the proof of Theorem 3 we obtain the following.

Theorem 4 *Given an instance of the k-median problem in the d-dimensional Euclidean space, and any fixed $m > 0$, there are randomized algorithms that compute a $(1 + O(\log(m)/m))$-approximation, in expectation, with worst-case running times $O(2^{O(m^{d-1})} n \log^{d+6} n)$ and $O(2^{O(m^{d-1})} \max\{k, \log^{d+1} n\} n \log^2 n)$ respectively.*

For any given accuracy $\varepsilon > 0$, we want $\ln(m)/m \leq c\varepsilon$, where m is the number of portals and c an appropriately small constant. Let us assume $c = 1$, since we can always decrease the given value of ε. Using the computer algebra package Maple [7] we obtain that for $m = \ln(1/\varepsilon)/(0.5\varepsilon)$, $\ln(m)/m \leq \varepsilon$ for any $\varepsilon \in (0, 0.5)$ while at the same time $m > 2$. On the other hand, setting $m = 3$, always yields an error less than 0.5. Therefore the asymptotic number of portals required for an accuracy of ε in the objective is

$$O(\log(1/\varepsilon)/\varepsilon).$$

We obtain the following reinterpretation of Theorem 4.

Theorem 5 *Given an instance of the k-median problem in the d-dimensional Euclidean space, and any fixed $\varepsilon > 0$, there are randomized algorithms that compute a $(1+\varepsilon)$-approximation, in expectation, with worst-case running times*

$$O\left(2^{O\left((\frac{\log(1/\varepsilon)}{\varepsilon})^{d-1}\right)} n \log^{d+6} n\right) \text{ and } O\left(2^{O\left((\frac{\log(1/\varepsilon)}{\varepsilon})^{d-1}\right)} \max\{k, \log^{d+1} n\} n \log^2 n\right).$$

5.4 Uncapacitated facility location

In this section we extend the results to uncapacitated facility location. Our structure theorems clearly hold as far as the service cost is concerned. We need to implement a dynamic programming algorithm that also keeps track of the facility cost. In order to be able to preprocess the points so that they lie on a polynomial-size grid we need the existence of a polynomial-factor approximation for the service cost as explained in Section 2. *The discussion below is predicated on such an estimate being available.*

The second approach for the k-median problem enumerated the service cost and computed for each service cost value the corresponding minimum number of facilities. The analogue here is to compute the minimum facility cost. This yields an algorithm that computes a $(1+\varepsilon)$-approximation in expectation with worst-case running time

$$O\left(2^{O\left(\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)^{d-1}\right)} n \log^{d+6} n\right).$$

Note that the latter time bound holds without any assumption on the range of the facility costs.

Acknowledgement. Many thanks to Sariel Har-Peled for valuable comments on a draft of this paper. The authors are grateful to Jan Remy for identifying a gap in the dynamic program and allowing us to include in the paper the fix he proposed.

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, 1999.

[2] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Journal of the ACM*, 45:753–782, 1998.

[3] S. Arora, P. Raghavan, and S. Rao. Polynomial time approximation schemes for the euclidean k-medians problem. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 106–113, 1998.

[4] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33:544–562, 2004.

[5] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[6] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[7] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual*. Springer-Verlag, 1991.

[8] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for group Steiner tree and k-median. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 114–123, 1998.

[9] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[10] M. Charikar, S. Guha, E. Tardos, and D. Shmoys. A constant factor approximation algorithm for the k-median problem. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 1–10, 1999.

[11] K. Chen. On k-median clustering in high dimensions. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 1177–1185, 2006.

[12] F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, editors, *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, volume 1412 of *LNCS*, pages 180–194. Springer-Verlag, Berlin, 1998.

[13] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory.* John Wiley and Sons, Inc., New York, 1990.

[14] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 434–444, 1988.

[15] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[16] S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.

[17] S. Har-Peled. Clustering motion. *Discrete Comput. Geom.*, 31(4):545–565, 2004.

[18] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 291–300, 2004.

[19] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[20] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48:274 – 296, 2001.

[21] S. G. Kolliopoulos and S. Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem. In J. Nešetřil, editor, *Proceedings of the 7th Annual European Symposium on Algorithms*, volume 1643 of *Lecture Notes in Computer Science*, pages 378–389. Springer-Verlag, 1999.

[22] J. H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44:245–249, 1992.

[23] J. H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

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

[25] S. Rao and W. D. Smith. Improved approximation schemes for geometrical graphs via *spanners* and *banyans*. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 540–550, 1998.

[26] J. Remy and A. Steger. Approximation schemes for node-weighted geometric Steiner tree problems. To appear in Algorithmica, special issue on APPROX/RANDOM 2005.

[27] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.