Exact and Approximation Algortihms for Clustering [*]

Pankaj K. Agarwal [†]   Cecilia M. Procopiuc [‡]

July 8, 1997

Abstract

In this paper we present a $n^{O(k^{1-1/d})}$ time algorithm for solving the k-center problem in $\mathbb{R}^d$, under L_∞ and L_2 metrics. The algorithm extends to other metrics, and can be used to solve the discrete k-center problem, as well. We also describe a simple $(1+\epsilon)$-approximation algorithm for the k-center problem, with running time $O(n \log k) + (k/\epsilon)^{O(k^{1-1/d})}$. Finally, we present a $n^{O(k^{1-1/d})}$ time algorithm for solving the L-capacitated k-center problem, provided that $L = \Omega(n/k^{1-1/d})$ or $L = O(1)$. We conclude with a simple approximation algorithm for the L-capacitated k-center problem.

[*] The work on this paper was partially supported by a National Science Foundation Grant CCR-93–01259, by an Army Research Office MURI grant DAAH04-96-1-0013, by a Sloan fellowship, by an NYI award and matching funds from Xerox Corporation, and by a grant from the U.S.-Israeli Binational Science Foundation.

[†] Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA

[‡] Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA

1 Introduction

Clustering a set of points into a few groups is frequently used for statistical analysis and classification in numerous applications, including information retrieval [6, 7, 8, 26], facility location [11, 30], data mining [2, 28], spatial data bases [9, 20, 25], data compression [23], image processing [19, 27], astrophysics [22], and scientific computing [5]. Because of such a diversity of applications, several variants of clustering problems have been proposed and widely studied. However, they all require a partition of a given set of points into clusters that optimizes a given objective function. In this paper we present efficient algorithms for a number of clustering problems.

Problem statement and our model. Let S be a set of n points in a d-dimensional metric space $(\mathbb{R}^d, \rho)$, and $k \leq n$ a positive integer. A *k-clustering* of S is a partition Σ of S into k subsets $S_1, \ldots, S_k$. Each S_i is called a *cluster* and k is called the *number of clusters.* We define the *size* of a cluster S_i to be the maximum distance (under the ρ-metric) between a fixed point c_i, called the *center* of the cluster, and a point of S_i. The size of a k-clustering Σ is the maximum size of a cluster in Σ. Such a clustering is called *center k-clustering* of size w.[1] The underlying metric ρ depends on the application.

The so-called *k-center* problem is defined as follows: *Given a set S of n points in a d-dimensional metric space $(\mathbb{R}^d, \rho)$ and an integer k, compute a k-clustering of S of the smallest possible size.* The k-center problem can be formulated as covering S by k congruent disks (under the ρ-metric) of the smallest possible size. If the centers of the clusters are required to be a subset of the input points, the problem is called the *discrete k-center problem.*

In some applications (e.g. facility location [11, 30], astrophysics [22]), not only the size of a cluster is important but also the number of points in the cluster. We can generalize the notion of k-clustering as follows. Given an integer $L > 0$, a *L-capacitated k-clustering* is a k-clustering in which there are at most L points in each cluster. The *capacitated k-center* problem is to compute a k-clustering of the smallest size so that each cluster has at most L points.

Previous results. There is a vast literature on clustering problems, see, for example, the books [3, 11, 18], the survey paper [1], and the references there in. Even the simplest clustering problems are known to be NP-Hard, including the Euclidean k-center problem in the plane [13, 24]. In fact, it is NP-Hard to approximate the two-dimensional k-center problem within a factor of < 2 even under the L_∞-metric [12]. The greedy algorithm by Gonzalez [14] gives a 2-approximation algorithm for the k-center problem in any metric space. This algorithm requires $O(kn)$ distance computations. The running time was improved by Feder and Greene [12] to $O(n \log k)$ for any L_p-metric. Several efficient algorithms have been developed for Euclidean and rectilinear k-center problems when k is small; see the survey by Agarwal and Sharir [1] for a summary of such results.

Gonzalez [15] presented an $n^{O(l)}$-time algorithm for the k-center problem in $\mathbb{R}^d$ under the L_∞ metric, when the points lie in a horizontal strip of height l; it can be extended to the L_2-metric as well. Hwang et al. [17] gave an $n^{O(\sqrt{k})}$-time algorithm for the Euclidean k-center problem in the plane. There algorithm is based on the following property: If S can be covered by k congruent

[1] Another commonly used definition of the size of a cluster S_i is the maximum distance between any two points of S_i. k-clustering of size w under this definition is called a *pairwise* k-clustering of size w.

disks of radius r, then there exist a set $\mathcal{D}$ of k disks of radius r and a line ℓ so that $\mathcal{D}$ covers S, each disk of $\mathcal{D}$ contains two points of S on its boundary, ℓ intersects $O(\sqrt{k})$ disks of $\mathcal{D}$, and at most $2k/3$ disks of $\mathcal{D}$ are contained in each of the halfplanes bounded by ℓ. Using this property, they develop a dynamic-programming based algorithm that determines in $n^{O(\sqrt{k})}$ time whether S can be covered by k disks of radius r. By doing a binary search, they compute an optimal k-clustering. This approach can be extended to the L_∞-metric as well [29]. In another paper, Hwang et al. [16] gave a $n^{O(\sqrt{k})}$ time algorithm that also works for the Euclidean discrete k-center problem.

Not much is known about the capacitated k-center problem. Bar-Ilan et al. [4] gave the first polynomial time approximation algorithm for the capacitated k-center problem, achieving an approximation factor of 10. They also proposed an approximation algorithm for the capacitated problem with fixed cluster size. The algorithm approximates the minimum number of clusters by a factor of $\lceil \ln n \rceil$. Khuller and Sussmann [21] improved the approximation factor for capacitated k-center to 6 (5 for a slightly different version of the problem). Recently Shmoys et al. [30] presented approximation algorithms for some generalizations of the capacitated k-center problem, using relaxation techniques.

Our results. One of the main results of this paper is an $n^{O(k^{1-1/d})}$-time algorithm for the k-center problem in $\mathbb{R}^d$, under any L_p-metric (Section 2). In fact our algorithm works for more general metrics, as discussed in the next section. For the sake of simplicity, we will describe the algorithm for the L_∞-metric in the plane. In this case our algorithm is based on following observation: If S can be covered by a set $\mathcal{C}$ of k congruent squares, then either S lies in a horizontal strip of width $\sqrt{k}+2$ or there exists a a horizontal line ℓ that intersects at most $\sqrt{k}$ squares of $\mathcal{C}$. Using this observation and the algorithm by Gonzalez [15], we develop a dynamic-programming based algorithm for finding an optimal solution. Our approach also yields an $n^{O(k^{1-1/d})}$-time algorithm for the discrete k-center problem in $\mathbb{R}^d$.

Recall that the algorithm by Hwang et al. [17] also runs in time $n^{O(\sqrt{k})}$ for $d=2$. Our algorithm is however not only considerably simpler than theirs, but the constant hidden in the big-Oh notation is smaller. Moreover, our algorithm is straightforward to extend to higher dimensions.

In Section 3 we describe a simple $(1+\epsilon)$-approximation algorithm for the k-center problem, whose running time is $O(n \log k) + (k/\epsilon)^{O(k^{1-1/d})}$. It is a combination of the greedy algorithm by Gonzalez and our algorithm for computing an optimal k-center.

Finally, we study the capacitated k-center problem in Section 4. Our exact algorithm does not extend to the capacitated k-center problem for all ranges of capacity (see Section 4 for a more detailed discussion), but we present an $n^{O(\sqrt{k})}$-time algorithm for computing an L-capacitated k-center under any L_p-metric, provided $L = O(1)$ or $L = \Omega(n/\sqrt{k})$. Our argument can be generalized to higher dimensions too. Finally, we present a simple approximation algorithm for the capacitated k-center problem.

2 The Exact Algorithm

In this section we describe a subexponential algorithm for computing the optimal k-center for a set S of n points in $\mathbb{R}^d$. For the sake of simplicity we describe the algorithm for the L_∞ metric, though it works for other metrics too. As noted above, the k-center problem under the L_∞ metric can be formulated as covering S by k congruent hypercubes of the smallest possible size.

Let σ^* be the size of the optimal k-clustering. We claim that $\sigma^* = d_\infty(p_i, p_j)/2$ for some $p_i, p_j \in S$. Indeed, if $\mathcal{C}$ is an optimal cover of S, then there exists a hypercube $C \in \mathcal{C}$ and two opposite facets f_1, f_2 of C so that f_1 and f_2 contain at least one point of S. Otherwise, we can shrink all hypercubes of size w^* and obtain a cover of S of smaller size. Hence, twice the size of an optimal k-clustering is one of the $O(n^2)$ distances (under the L_∞ metric) between any two of the input points. The algorithm performs a binary search on the set of these distances, testing at each step whether S can be covered by k congruent hypercubes of size σ, for some given $\sigma > 0$.

A similar approach can be employed when the metric is L_2. In this case, the clusters are balls, and there exists a k-clustering of S of smallest size so that each ball in the cluster is defined by at most $d+1$ points of S [10]. Then, σ^* is the radius of one of the balls defined by $\leq d+1$ input points. There are $O(n^{d+1})$ radii of balls determined by $\leq d+1$ points of S, and we can compute each one in time $O(d^3)$.

It thus suffices to describe an algorithm for the decision problem. We present the decision procedure for the planar case first, under the L_∞ metric, and then discuss how to extend our algorithm to the L_2 metric and to higher dimensions.

2.1 The Decision Procedure in $\mathbb{R}^2$

Assume without loss of generality that $\sigma = 1/2$. In this case, the problem is to decide whether S can be covered by k unit squares, and if so, to return such a cover (by unit squares, we mean squares of sidelength 1). We assume that the x- and y-coordinates of all points are distinct. In the full version, we will discuss how to modify the algorithm to handle points in degenerate position.

Definition 2.1 The *index* of a set of squares $\{C_1, \ldots, C_q\},\ q \leq n$ is the $2n$-vector

$$(\infty, \ldots, \infty, x_1, \ldots, x_q, \infty, \ldots, \infty, y_1, \ldots, y_q),$$

where (x_i, y_i) is the lower left corner of the square C_i, $1 \leq i \leq q$, and $x_1 \leq x_2 \leq \ldots \leq x_q$ (x_1 is at position $n-q+1$, and y_1 is at position $2n-q+1$). Given a set of n points S, the *optimal cover* of S by squares of size 1 is the cover $\mathcal{C}^*$ whose index is maximal in the lexicographic order.

An immediate observation is that the optimal cover of S has the minimum number of squares among all the covers of S. By definition, the optimal cover is unique. We will also use the word *subcover* to denote a set of squares that cover some subset of S.

Definition 2.2 A square of $\mathcal{C}$ is called *anchored* if it has at least one input point on its left side and one (not necessarily different) input point on its bottom side. We call the two points the *anchors* of the square. Let $\mathcal{C}$ be a cover of S. We say that $\mathcal{C}$ is an *anchored cover* if all of its squares are anchored, and moreover, the anchors of each square are not covered by the any other square of $\mathcal{C}$ (see Figure 1).

We denote by Γ the set of all anchored unit squares. Following a similar argument to the one

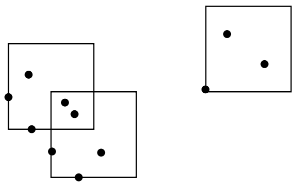

Figure 1: An anchored cover of S

in [15], we can prove the following lemma.

Lemma 2.3 *The optimal cover of S is an anchored cover.*

Given an integer k, our algorithm will either compute the optimal cover (if the number of squares in the optimal cover is at most k), or it will return that S cannot be covered by k squares. The algorithm is based on the following property of the optimal cover, which we prove in the lemma below: there exists a set of horizontal lines that divide the plane into strips, so that any strip that contains input points has height at most $\sqrt{k}+2$, and so that each of these lines intersects at most $\sqrt{k}$ squares of the optimal cover. We compute the optimal covers on these strips and then take their union. We show that the lines can be chosen out of a finite set of lines, and we use dynamic programming. To compute the optimal cover on a "thin" strip, we use a slightly modified version of the algorithm by Gonzalez [15], which we refer to as the *Strip Algorithm.* We present the main ideas of this algorithm below.

Strip Algorithm: Let S be a set of n points lying in a horizontal strip of height l. Gonzalez describes a sweep line algorithm for computing a cover of S by the minimum number of squares. His algorithm relies on the following lemma, whose proof follows from a simple packing argument.

Lemma 2.4 *Every vertical line intersects at most $2l-1$ squares of the optimal cover of S.*

The algorithm sweeps the plane from left to right and maintains a family $\mathcal{F} = \{\mathcal{C}_1, \ldots, \mathcal{C}_u\}$ with the following properties:

(i) Each $\mathcal{C}_i$ is a subset of Γ, and it covers all points lying to the left of the sweep line.

(ii) Let $\mathcal{C}'$ be the subset of squares in the optimal cover of S that do not lie entirely to the right of the sweep line. Then $\mathcal{C}' \in \mathcal{F}$.

(iii) For each i, $1 \le i \le u$, the subset of squares of $\mathcal{C}_i$ intersected by the sweep line is distinct.

Condition (iii) and Lemma 2.4 imply $u = O(n^{4l-2})$. Whenever the sweep line passes through a point of S, the algorithm updates the set $\mathcal{F}$ so that conditions (i)-(iii) are satisfied. See the original paper [15] for details. We can modify the algorithm so that it returns the optimal cover of S in the sense of our definition.

Handling thick strips: To describe how we generate the optimal cover when the points do not lie in a strip of bounded height, we need a few more definitions. Let H be the set of at most

$2n+1$ horizontal lines defined as

$$H = \{y = y(p) \mid p \in S\} \cup \{y = y(p) + 1 \mid p \in S\} \cup \{y = y_{min} - 1\},$$

where $y(p)$ denotes the y-coordinate of the point p, and y_{min} is the minimum among $y(p),\ p \in S$. Note that each horizontal edge of a square in Γ is contained in a line in H. Therefore, for any $X \subseteq \Gamma$, any horizontal lines lying between two consecutive lines of H intersect the same squares of X. The lines that divide the plane, as discussed above, will be chosen only from the set H.

Lemma 2.5 *Let S be a set of n points lying in a horizontal strip w, and let $\mathcal{C}$ be the optimal cover of S with at most k squares. If the height of w is at least $\sqrt{k}+2$, and no square of $\mathcal{C}$ intersects the boundary of w, then there exists a line $h \in H$ lying in the interior of w that intersects at most $\sqrt{k}$ squares of $\mathcal{C}$.*

Proof: Let ℓ_1 and ℓ_2 denote the two horizontal lines that form the boundary of w. Fix an arbitrarily small parameter ϵ and divide w by equidistant horizontal lines $h_1, \ldots, h_r$, so that the distance between h_i and h_{i+1} is $1+\epsilon$, the distance between ℓ_1 and h_1 is ϵ, and the distance between h_r and ℓ_2 is at most $1+\epsilon$. Then, no square intersecting h_i can intersect h_{i+1}. Suppose that each h_i intersects more than $\sqrt{k}$ squares of $\mathcal{C}$. Then, $r < \mathcal{C}/\sqrt{k} \leq \sqrt{k}$, and the height of the strip is at most $\epsilon + (1+\epsilon)r < \sqrt{k} + \epsilon(1+\sqrt{k})$. Choosing ϵ sufficiently small, we obtain a contradiction with the fact that w has height $> \sqrt{k}+2$.

Hence, there exists an h_i that intersects at most $\sqrt{k}$ squares of $\mathcal{C}$. If $h_i \in H$, choose $h = h_i$. Otherwise, let h_-, h_+ be the horizontal lines of H that lie immediately below and above h_i. Clearly, h_-, h_+ lie in w. It is easy to check that h_- and h_i intersect the subset of $\mathcal{C}$. If $h_- \neq \ell_1$, choose $h = h_-$. If $h_- = \ell_1$, then h_- does not intersect any square of $\mathcal{C}$, and so h_+ intersects at most one square of $\mathcal{C}$. Moreover, $h_+ \neq \ell_2$ (otherwise, $S = \emptyset$), so we can choose $h = h_+$. □

Remark 2.6 (i) If the points are in general position, the lemma works even if the height of w is bigger than $\sqrt{k}$. But we state it as above in anticipation of the result for points in degenerate position.

(ii) The proof holds for any cover of S by at most k unit squares.

Definition 2.7 (i) For any $h \in H$ and $D \subseteq \Gamma$, we say that (h, D) is a *canonical pair* if all squares in D intersect h, and $0 \leq |D| \leq \sqrt{k}$.

(ii) We define a *c-strip* to be the triple $\tau = (w, A, B)$, where w is a horizontal strip with lower boundary ℓ_1 and upper boundary ℓ_2, so that (ℓ_1, A) and (ℓ_2, B) are canonical pairs. Let $S_\tau \subseteq S$ be the set of input points that lie in w, but not in any square of $A \cup B$. We define the *optimal cover associated with* τ, $\mathcal{C}_\tau$, to be the optimal cover of S_τ.

(iii) A c-strip τ is *legal* if no square of $\mathcal{C}_\tau$ intersects the boundary of w. Otherwise, τ is *illegal.*

Using Lemma 2.5, we obtain the following result (see Appendix for a proof).

Lemma 2.8 *Let $\tau = (w, A, B)$ be a legal c-strip of height bigger than $\sqrt{k}+2$. If $S_\tau \neq \emptyset$ and $|\mathcal{C}_\tau| \leq k$, there exists a canonical pair (h, D) so that h divides w into two strips w_-, w_+, each of* *height strictly less than the height of w, and so that $\tau_- = (w_-, A, D)$ and $\tau_+ = (w_+, D, B)$ are legal c-strips. Moreover, $\mathcal{C}_\tau = \mathcal{C}_{\tau_-} \cup D \cup \mathcal{C}_{\tau_+}$.*

Decision Algorithm. In the following, we say that a c-strip τ is *thin* if the height of w is at most $\sqrt{k} + 2$. Otherwise, τ is *thick*. Lemma 2.8 suggests the following recursive algorithm. Start with a legal strip τ_B that contains all input points. If τ_B is a thin strip, compute $\mathcal{C}_{\tau_B}$ using the strip algorithm by Gonzalez. Otherwise, divide τ_B into two smaller legal strips, by a canonical pair (h, D), and solve the problem recursively on each one of the strips. Instead of trying all possibilities of dividing a large legal c-strip, we use dynamic programming and work our way up the recursion tree, starting from the leaves that correspond to thin strips. We now describe the algorithm in more detail.

The algorithm builds a table, each of whose entry is indexed by a c-strip $\tau = (w, A, B)$. If τ is legal and $|\mathcal{C}_\tau| \leq k$, the entry stores $k(\tau) = |\mathcal{C}_\tau|$ and the index $I(\tau)$ of $\mathcal{C}_\tau$; otherwise, $k(\tau)$ is set to ∞ and $I(\tau)$ is undefined. Fill the entry τ as follows: If $S_\tau = \emptyset$, then $k(\tau) = 0$ and $I(\tau)$ has ∞ on all positions. If τ is a thin strip, compute $\mathcal{C}_\tau$ using the strip algorithm for S_τ. If any square of $\mathcal{C}_\tau$ intersects a boundary of w, or $|\mathcal{C}_\tau| > k$, set $k(\tau) = \infty$; otherwise, store $k(\tau)$ and $I(\tau)$. Finally, if τ is a thick strip, compute all canonical pairs (h, D) for which h lies inside w. For each canonical pair (h, D), let w_- (respectively w_+) be the portion of w lying below (respectively above) h. Let

$$k(\tau) = \min_{(h,D)} (k(\tau_-) + k(\tau_+) + |D|).$$

If (h^*, D^*) is the pair for which this minimum is attained, and $k(\tau) \neq \infty$, then $I(\tau)$ is the index of $\mathcal{C}_{\tau_-^*} \cup D^* \cup \mathcal{C}_{\tau_+^*}$ (τ_-^*, τ_+^* are the two c-strips determined by (h^*, D^*)). This index can be computed in $O(k)$ time from $I(\tau_-^*)$ and $I(\tau_+^*)$. If there is a tie for the minimum, always choose the canonical pair that maximizes $I(\tau)$.

Let w_B be the strip bounded by the lowest and highest lines of H. The entry corresponding to $\tau_B = (w_B, \emptyset, \emptyset)$ gives the size and the index of the optimal cover of S. The running time can be proved to be $O(n^{8\sqrt{k}+10})$. The correctness of the algorithm follows from Lemma 2.8 and the rules for computing $k(\tau)$ and $I(\tau)$. Putting everything together, we conclude:

Theorem 2.9 *If S is a set of n points in the metric space $(\mathbb{R}^2, L_\infty)$, we can find the optimal k-clustering of S in time $O(n^{8\sqrt{k}+10})$.*

If the underlying metric is L_2, the decision problem becomes whether S can be covered by k circles of diameter 1. Using arguments similar to the above, we can still prove a claim similar to Lemma 2.5, and modify the algorithm in a straightforward manner. Although the constants change, the table size and the Strip Algorithm are of the same order as before. Omitting all the details, we conclude the following

Theorem 2.10 *If S is a set of n points in the metric space $(\mathbb{R}^2, L_2)$, we can find the optimal k-clustering of S in time $n^{O(\sqrt{k})}$.*

Remark 2.11 The algorithm can be modified to solve the discrete k-center problem, as well. The techniques presented in this paper extend to any metric ρ for which the following conditions are met: the unit disk under the metric ρ has constant complexity and constant aspect ratio, and a unit square can be covered by a constant number of unit disks under the metric ρ. In particular, the algorithm works under any L_p metric.

2.2 The Decision Procedure in $\mathbb{R}^d$

The main ingredients we need for computing the exact cover in $\mathbb{R}^d$ are the Strip Algorithm in $\mathbb{R}^d$ and an extension of Lemma 2.5 to higher dimensions. We define a strip to be the intersection of $2d-2$ halfspaces of the form $x_i \geq a_1$ and $x_i \leq b_i$, for $1 \leq i \leq d-1$, where $a_i, b_i \in \mathbb{R}$. The notion of anchored square can be extended in a natural way (note that we need d anchors per hypercube). The set H will contain hyperplanes orthogonal to any of the first $d-1$ directions, so that a hyperplane orthogonal to the ith direction contains an input point $(x_1, \ldots, x_i, \ldots, x_d)$, or the point $(x_1, \ldots, x_i+1, \ldots, x_d)$, or is so that S lies strictly in one of its halfspaces.

Lemma 2.4 can be generalized as follows: when the points lie in a strip of width at most l in each of the first $d-1$ directions, any hyperplane of equation $x_d = a$, $a \in \mathbb{R}$, intersects at most $2l^{d-1}-1$ hypercubes of the optimal cover. The generalization of Lemma 2.5 is as follows: if the points lie in a strip of width $\Omega(k^{1/d})$ in some direction $i \leq d-1$, and the optimal cover of the points does not intersect the boundaries of the strip, then there exists a hyperplane $h \in H$ in the interior of the strip, that is orthogonal to direction i and intersects $O(k^{1-1/d})$ hypercubes of the optimal cover.

We obtain the following result.

Theorem 2.12 *If S is a set of points in the metric space $(\mathbb{R}^d, L_\infty)$, we can find the optimal k-clustering of S in time $n^{O(k^{1-1/d})}$.*

3 The Approximation Algorithm

Even for moderate values of n, the algorithm of the previous section is not practical. However, we can obtain a faster algorithm if we accept an approximate solution for the k-center problem. More precisely, we want to solve the following problem: *Given a set S of n points, an integer $k \leq n$, and a real parameter ϵ, $0 < \epsilon < 1$, compute a k-clustering of S of size $(1+\epsilon)\sigma^*$, where σ^* is the size of an optimal k-clustering of S.*

We present the algorithm for the L_∞ metric; it can also be adapted for any L_p-metric. First, compute a real number σ_0, so that $\sigma^* \leq \sigma_0 \leq 2\sigma^*$, using the algorithm by Feder and Greene [12], in time $O(n \log k)$. Let Z be a grid of size $\delta = \sigma_0\epsilon/3$, i.e. $Z = \{(i\delta, j\delta) \mid i, j \in \mathbb{Z}\}$. Let P be the set of grid points v, so that at least one of the grid cells adjacent to v contains a point of S. See Figure 2.

Next, compute the optimal cover of P, using the algorithm described in the previous section. Suppose the algorithm returns $\mathcal{C} = \{C_1, \ldots, C_k\}$ as optimal cover of P, and that the size of each hypercube C_i is σ_1. For each $C_i \in \mathcal{C}$, let C_i' be the hypercube with the same center as C_i and of size $\sigma_2 = \sigma_1 + \sigma_0\epsilon/6$. Return $\mathcal{C}' = \{C_1', \ldots, C_k'\}$.

The proof of correctness of the above algorithm follows from the following lemma, whose proof

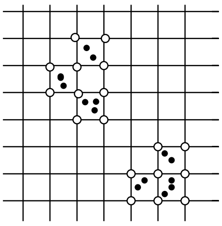

Figure 2: The set of grid vertices P

is given in the appendix.

Lemma 3.1 *(i) $\mathcal{C}'$ is a cover of S.*

(ii) $\sigma_2 \le (1+\epsilon)\sigma^*$.

An easy calculation shows that $|P| = O(k/\epsilon^d)$, so we obtain the following result.

Theorem 3.2 *Given a set S of n points in the metric space $(\mathbb{R}^d, L_\infty)$, we can find a cover of S of cluster size $(1+\epsilon)w^*$ in time $O(n \log k + (k/\epsilon)^{O(k^{1-1/d})})$.*

4 The Capacitated k-Center Problem

In this section, we propose a sub-exponential algorithm for the capacitated k-center problem, provided that L, the maximum number of points in a cluster, is either $\Omega(n/\sqrt{k})$ or $O(1)$. We assume that the points are in general position. The overall approach is similar to that in Section 2. For simplicity, we describe the algorithm in $\mathbb{R}^2$ under the L_∞ metric. In this case, the decision problem for the capacitated k-center can be formulated as follows: *Let S be a set of n points in the plane, and k, L be two positive integers. Do there exist k unit squares (not necessarily distinct), so that each point of S can be assigned to one of the squares containing it in such a way that each square is assigned at most L points?* If the answer is "yes", the algorithm should return a multiset $\mathcal{C}$ of at most k squares and an assignment function $\chi : S \to \mathcal{C}$ that satisfies the required conditions. We will therefore denote the output as $(\mathcal{C}, \chi)$. Throughout this section, by a *cover* of S we will mean a L-capacitated cover of S with unit squares. We will use the term *smallest cover* to denote a cover with minimum number of squares.

In general, the decision algorithm does not extend to capacitated k-clustering. Although Lemma 2.5 and the correspondent of Lemma 2.8 can still be proved, Lemma 2.4 required for Gonzalez's algorithm is not true for arbitrary values of L. In particular, we can construct a counterexample of a set of n points lying in a horizontal strip of height l, so that for any smallest cover $(\mathcal{C}, \chi)$ of S, there are vertical lines that intersect $\Omega(L \cdot l)$ squares of $\mathcal{C}$ (see Lemma A.1 in the appendix).

We can however prove Lemma 2.4 for some restricted values of L, namely when $L = \Omega(n/l)$ or $L = O(1)$. We first state the result for $L = \Omega(n/l)$ (see Appendix for a proof).

Lemma 4.1 *Let S be a set of n points lying in a horizontal strip of height l, and L be a natural number. If $L = \Omega(n/l)$, then any vertical line intersects $O(l)$ squares of a smallest cover of S.*

A more interesting situation occurs when $L = O(1)$. In general, a vertical line may intersect up to $k = \Omega(n)$ squares of a smallest cover, but we will prove a slightly weaker result that is sufficient to extend Gonzalez's algorithm. For any cover $(\mathcal{C}, \chi)$ of S, we say that a square C of $\mathcal{C}$ is *active* with respect to a vertical line ℓ if there exist two points p, q assigned to C, so that p lies to the left of ℓ, or on ℓ, and q lies strictly to the right of ℓ. Otherwise, C is *inactive* with respect to ℓ.

Lemma 4.2 *Let S be a set of n points that lie in a horizontal strip of height l, and let L be a constant number. There exists a smallest cover $(\mathcal{C}, \chi)$ of S, so that there are at most $O(l)$ active squares of $\mathcal{C}$ with respect to any vertical line.*

We prove the lemma using the following two results (see appendix for proof of Lemma 4.3).

Lemma 4.3 *Let S be a set of n points lying in a unit square, and let $1 \leq L \leq n$ be an integer. There exists a smallest cover $(\mathcal{C}, \chi)$ of S so that at most one square of $\mathcal{C}$ is active with respect to any vertical line.*

Lemma 4.4 *Let $S_1, \ldots, S_u$ be a family of pairwise disjoint sets of points in the plane, so that each S_i lies in a unit square U_i. There exists a smallest cover $(\mathcal{C}, \chi)$ of $S = \bigcup_{i=1}^{u} S_i$, so that at most L^{u+1} squares are assigned points belonging to more than one set.*

Proof: We prove the claim for $u = 2$. The proof can be generalized to any u. Let $(\mathcal{C}, \chi)$ be a smallest cover of S, with the minimum number of squares having points assigned from more than one set.

For $1 \leq i, j \leq L-1$, so that $i+j \leq L-1$, let $\mathcal{M}_{ij} = \{C \in \mathcal{C} \mid |C \cap S_1| = i \text{ and } |C \cap S_2| = j\}$. We claim that $|\mathcal{M}_{ij}| < L$. Indeed, if $|\mathcal{M}_{ij}| \geq L$, then let $C_1, \ldots, C_L$ be L squares of $\mathcal{M}_{ij}$. Let $X_i \subseteq S_i$, $i \in \{1, 2\}$, be the set of points assigned to $C_1, \ldots, C_L$. Obviously, $|X_1| = iL$, $|X_2| = jL$. Replace $C_1, \ldots, C_L$ in $\mathcal{C}$ by i copies of U_1 and j copies of U_2, and assign L points of X_1 (respectively X_2) to each copy of U_1 (respectively U_2). If $i + j < L$, we obtain a smaller cover of S; and if $i + j = L$, we obtain a smallest cover of S with fewer squares having points assigned from both S_1 and S_2, a contradiction. The lemma is now immediate. □

Proof of Lemma 4.2 Cover S by a family $\mathcal{U}$ of pairwise disjoint unit squares. For each $U_i \in \mathcal{U}$, let $S_i \subseteq S$ be the set of points contained in U_i. We say that two squares $U_i, U_j \in \mathcal{U}$ are neighbours if the minimum distance between S_i and S_j is at most 1. Since the squares in $\mathcal{U}$ are pairwise disjoint, any square U_i has at most 8 neighbours. Note that, if a square $C \in \mathcal{C}$ is assigned points from a set S_i, as well as from some other sets, then any point assigned to C is either contained in U_i, or in some neighbour of U_i.

Using this observation and the result of Lemma 4.4, it is not difficult to prove that there exists a smallest cover $(\mathcal{C}, \chi)$ of S so that, for each set S_i, at most L^9 squares of $\mathcal{C}$ are assigned points from S_i and from at least one other set.

We construct another L-capacitated cover $(\mathcal{C}', \chi')$ as follows: Let $\mathcal{C}^* \subseteq C$ be the set of squares to which points belonging to at least two sets are assigned. Let S_i' denote the points of S_i that are not assigned to any square in $\mathcal{C}^*$, and $(\mathcal{C}_i, \chi_i)$ be a cover of S'_i satisfying Lemma 4.3. Set $\mathcal{C}' = (\bigcup \mathcal{C}_i) \cup \mathcal{C}^*$ and let $\chi'(p) = \chi(p)$ if $\chi(p) \in \mathcal{C}^*$, and $\chi'(p) = \chi_i(p)$ if $p \in S'_i$. It is easy to see that $(\mathcal{C}', \chi')$ is a smallest L-capacitated cover of S.

We claim that there are at most $O(l)$ active squares of $\mathcal{C}$ with respect to any vertical line. Indeed, let ℓ be a vertical line, and R_ℓ be the set of all points in the strip within distance 1 from ℓ (R_ℓ is a rectangle of height l and width 2). We define $\mathcal{U}_\ell \subseteq \mathcal{U}$ as the subset of squares that intersect R_ℓ. Since the squares in $\mathcal{U}$ are pairwise disjoint, $|\mathcal{U}_\ell| = O(l)$. Any square $C \in \mathcal{C}'$ intersected by ℓ lies in R_ℓ and all the points assigned to C lie in $\mathcal{U}_\ell$. Hence, at most $L^9 \cdot |\mathcal{U}_\ell|$ squares of $\mathcal{C}'$ are assigned points from at least two sets, and are intersected by ℓ. The way we obtained $(\mathcal{C}', \chi')$ ensures that the line ℓ intersects at most $|\mathcal{U}_\ell|$ squares of $\mathcal{C}'$ that are assigned points from only one set. The total number of squares intersected by ℓ is thus at most $(L^9 + 1) \cdot |\mathcal{U}_\ell| = O(l)$. □

To generate the optimal cover on a c-strip, we follow the approach of the Strip Algorithm, but we replace condition (iii) by

(iii') If $A_i(\ell)$ denotes the set of active squares of $\mathcal{C}_i$ with respect to the sweep line ℓ, then $A_i(\ell) \neq A_j(\ell)$, for any $1 \leq i < j \leq u$.

Next, we modify our dynamic programming algorithm to obtain the following

Theorem 4.5 *Given a set S of n points in the plane, and integers k and L, the optimal solution of the capacitated k-center problem for S, with load constraint L, can be obtained in time $n^{O(\sqrt{k})}$ if one of the following is true: $L = \Omega(n/\sqrt{k})$, or $L = O(1)$.*

We conclude by presenting an approximation algorithm for the capacitated k-center problem. Given the set of points S, we define a $(c_1 k, c_2 L, c_3 \sigma^*)$ cover of S as a cover that uses at most $c_1 k$ clusters, each containing at most $c_2 L$ points, and of cluster size at most $c_3 \sigma^*$, where σ^* is the cluster size of the optimal solution. Khuller and Sussmann [21] gave a polynomial algorithm for computing a $((2/c)k, cL, 2\sigma^*)$ cover, for any $c > 1$.

We present an algorithm that computes a $(\frac{2}{1+\eta}k, (1+\eta)L, (1+\epsilon)\sigma^*)$ cover of S, for any ϵ, η so that $0 < \epsilon, \eta < 1$. The algorithm proceeds as follows: Compute a cover $\mathcal{C}$ that is an $(1+\epsilon)$-approximation of the optimal uncapacitated cover of S. For each $C \in \mathcal{C}$, let *unassigned*(C) be the number of points covered by C and not yet assigned to any square (initially, all points are unassigned). Let *unassigned*$(C) = qL + \alpha$, $0 \leq \alpha \leq L - 1$. If $\alpha \leq \eta L$, create $q - 1$ copies of C, else create q copies of C, and assign the *unassigned*(C) points evenly to C and its copies. Return the union of all these squares, together with the assignment.

Theorem 4.6 *Given a set S of n points in the plane, and k, L two natural numbers, we can obtain a $(\frac{2}{1+\eta}k, (1+\eta)L, (1+\epsilon)\sigma^*)$ cover of S in time $O(n \log k + (k/\epsilon)^{O(\sqrt{k})})$, for any $0 < \epsilon, \eta < 1$.*

References

[1] P. K. Agarwal and M. Sharir, Efficient algorithms for geometric optimization, Tech. Rep. CS-1996-19, Dept. Computer Science, Duke University, 1996.

[2] R. Agrawal, A. Ghosh, T. Imielinski, B. Iyer, and A. Swami, An interval classifier for database mining applications, *Proceedings of the 18th Conference on Very Large Databases*, Morgan Kauffman, August 1992.

[3] M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, 1973.

[4] J. Bar-Ilan, G. Kortsarz, and D. Peleg, How to allocate network centers, *J. Algorithms*, 15 (1993), 385–415.

[5] M. Berger and I. Rigoutsos, An algorithm for point clustering and grid generation, *IEEE Trans. Systems, Man and Cybernetics.*, 21 (1991), 1278–1286.

[6] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, Incremental clustering and dynamic information retrieval, *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997, pp. 626–634.

[7] D. R. Cutting, D. R. Karger, and J. O. Pedersen, Constant interaction-time scatter/gather browsing of very large document collections, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993, pp. 126–134.

[8] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey, Scatter/gather: A cluster-based approach to browsing large document collections, *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992, pp. 318–329.

[9] A. A. Diwan, S. Rane, S. Seshadri, and S. Sudarshan, Clustering techniques for minimizing external path length., *Proceedings of the International Conference on Very Large Databases*, Morgan Kauffman, 1996.

[10] Z. Drezner, The p-centre problems — Heuristic and optimal algorithms, *J. Oper. Res. Soc.*, 35 (1984), 741–748.

[11] Z. Drezner, ed., *Facility Location*, Springer-Verlag, New York, 1995.

[12] T. Feder and D. H. Greene, Optimal algorithms for approximate clustering, *Proc. 20th Annu. ACM Sympos. Theory Comput.*, 1988, pp. 434–444.

[13] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Inform. Process. Lett.*, 12 (1981), 133–137.

[14] T. Gonzalez, Clustering to minimize the maximum intercluster distance, *Theoret. Comput. Sci.*, 38 (1985), 293–306.

[15] T. Gonzalez, Covering a set of points in multidimensional space, *Inform. Process. Lett.*, 40 (1991), 181–188.

[16] R. Z. Hwang, R. C. Chang, and R. C. T. Lee, The generalized searching over separators strategy to solve some NP-Hard problems in subexponential time, *Algorithmica*, 9 (1993), 398–423.

[17] R. Z. Hwang, R. C. T. Lee, and R. C. Chang, The slab dividing approach to solve the Euclidean p-center problem, *Algorithmica*, 9 (1993), 1–22.

[18] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[19] J. Jolion, P. Meer, and S. Batauche, Robust clustering with applications in computer vision, *IEEE Trans. Pattern Analysis Mach. Intell.*, 13 (1991), 791–802.

[20] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

[21] S. Khuller and Y. J. Sussmann, The capacitated k-center problem, *Proceedings of the 4th Annual European Symposium on Algorithms*, 1996, pp. 152–166.

[22] R. Lupton, F. M. Maley, and N. Young, Data collection for the sloan digital sky survey: A network-flow heuristic, *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM/SIAM, January 1996, pp. 296–303.

[23] J. Makhoul, S. Roucos, and H. Gish, Vector quantization in speech coding, *Proceedings of the IEEE*, 73 (1985), 1551–1588.

[24] N. Megiddo and K. J. Supowit, On the complexity of some common geometric location problems, *SIAM J. Comput.*, 13 (1984), 182–196.

[25] R. T. Ng and J. Han, Efficient and Effective Clustering Methods for Spatial Data Mining, *Proceedings of the Twentieth International Conference on Very Large Databases*, 1994, pp. 144–155.

[26] P. Raghavan, Information retrieval algorithms: A survey, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1997, pp. 11–18.

[27] P. Schroeter and J. Bigün, Hierarchical image segmentation by multi-dimensional clustering and orientation-adaptive boundary refinement, *Pattern Recognition.*, 28 (1995), 695–709.

[28] J. Shafer, R. Agrawal, and M. Mehta, Sprint: A scalable parallel classifier for data mining., *Proceedings of the International Conference on Very Large Databases*, Morgan Kauffman, 1996.

[29] M. Sharir, Private communication, (1997).

[30] D. Shmoys, E. Tardos, and K. Aardal, Approximation algorithms for facility location problems, *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997, pp. 265–274.

A Appendix: Proofs of Lemmata

Lemma 2.8 Let $\tau = (w, A, B)$ be a legal c-strip of height bigger than $\sqrt{k} + 2$. If $S_\tau \neq \emptyset$ and $|\mathcal{C}_\tau| \leq k$, there exists a canonical pair (h, D) so that h divides w into two strips w_-, w_+, each of height strictly less than the height of w, and so that $\tau_- = (w_-, A, D)$ and $\tau_+ = (w_+, D, B)$ are legal c-strips. Moreover, $\mathcal{C}_\tau = \mathcal{C}_{\tau_-} \cup D \cup \mathcal{C}_{\tau_+}$.

Proof: By Lemma 2.5, there exists a line $h \in H$ that intersects at most $\sqrt{k}$ squares of $\mathcal{C}_\tau$. Let D denote the set of squares of $\mathcal{C}_\tau$ intersected by h. Then $\tau_- = (w_-, D, B)$ and $\tau_+ = (w_+, A, D)$ are c-strips. Let $\mathcal{C}_1$ (respectively $\mathcal{C}_2$) be the set of squares of $\mathcal{C}_\tau$ that lie in the interior of w_+ (respectively w_-). It can be shown that $\mathcal{C}_1$ (respextively $\mathcal{C}_2$) is the optimal cover of S_{τ_-} (respectively S_{τ_+}). Since τ is a legal c-strip, no square of $\mathcal{C}_1 \cup \mathcal{C}_2$ intersects the boundaries of τ. Therefore, τ_+ and τ_- are legal c-strips.

Obviously, $\mathcal{C}_\tau = \mathcal{C}_1 \cup D \cup \mathcal{C}_2 = \mathcal{C}_{\tau_-} \cup D \cup \mathcal{C}_{\tau_+}$. □

Lemma 3.1 (i) $\mathcal{C}'$ is a cover of S.

(ii) $\sigma_2 \leq (1+\epsilon)\sigma^*$.

Proof: (i) For any $p \in S$, there exists a point $v \in P$ so that $d_\infty(p,v) \leq \sigma_0\epsilon/6$. Indeed, p lies in a grid cell of size $\sigma_0\epsilon/3$, and so at least one vertex v of the cell is at a distance $\leq \sigma_0\epsilon/6$ of p (under the L_∞ metric). By construction, $v \in P$. If C_i is a hypercube of $\mathcal{C}$ that covers v, then C_i' covers p.

(ii) We first prove that $\sigma_1 \leq (1+2\epsilon/3)\sigma^*$. Let $\mathcal{C}^*$ be the optimal cover of S. For each hypercube $C_i^* \in \mathcal{C}^*$, let C_i^ℓ be the hypercube of size $\sigma^* + \sigma_0\epsilon/3$, with the same center as C_i^*, but of size . We claim that $\mathcal{C}^\ell = \{C_1^\ell, \ldots, C_k^\ell\}$ is a cover of P. Indeed, for any $v \in P$, there exists $p \in S$ so that p lies in a cell adjacent to v (by construction). Then, $d_\infty(v,p) \leq \sigma_0\epsilon/3$. If C_i^* is a hypercube of $\mathcal{C}^*$ that covers p, then C_i^ℓ covers v. Because $\mathcal{C}$ is the optimal cover of P, we obtain $\sigma_1 \leq \sigma^* + \sigma_0\epsilon/3 \leq (1+2\epsilon/3)\sigma^*$. Hence, $\sigma_2 = \sigma_1 + \sigma_0\epsilon/6 \leq (1+\epsilon)\sigma^*$. □

Lemma 4.1 Let S be a set of n points lying in a horizontal strip of height l, and L be a natural number. If $L = \Omega(n/l)$, then any vertical line intersects $O(l)$ squares of a smallest cover of S.

Proof: Let $L \geq A \cdot n/l$, for some constant $A \leq 1$. The argument follows the ideas in [15]. It is sufficient to show that all the points lying in a rectangle of height l and width 2 can be assigned to at most $(2+1/A)l - 1$ squares. Indeed, we need at most $2l$ squares to cover all the points in the rectangle. If some square covers $L' > L$ points, we assign L of the points to that square, and create a new square that covers the remaining $L' - L$ points. We repeat this procedure until all squares satisy the load constraint. Because $L \geq A \cdot n/l$, we create at most $l/A - 1$ new squares, for a total of $(2+1/A)l - 1$. □

Lemma 4.3 Let S be a set of n points lying in a unit square, and let $1 \leq L \leq n$ be an integer. There exists a smallest cover $(\mathcal{C}, \chi)$ of S so that at most one square of $\mathcal{C}$ is active with respect to any vertical line.

Proof: Let the points of S be ordered in the increasing order of their x-coordinates. Assign the first L points to a square C_1, the next L points to a square C_2, and so on. Because all points lie in a unit square, it is always possible to construct the squares C_i. Let $(\mathcal{C}, \chi)$ be the cover of S obtained by this method. Clearly, $(\mathcal{C}, \chi)$ is a smallest cover, and any vertical line intersects at most one active square of $\mathcal{C}$ (remember that the points are in general position). □

Lemma A.1 *There exists a set S of n points lying in a horizontal strip of height l, so that for any smallest cover $(\mathcal{C}, \chi)$ of S, there exists a vertical line that intersects $\Omega(L \cdot l)$ squares of $\mathcal{C}$.*

Proof Sketch Consider $\Omega(l)$ groups of L squares as in Figure 4. Each square has one point in its upper left corner, and $L-1$ points very close to its lower right corner (represented as the small black rectangle). All squares lie in a vertical strip of height l and width 2. The groups are suffficiently far from each other, so that no unit square can cover points from two distinct groups. The clustering in the figure is the only smallest capacitated clustering of the points, and there exists a vertical line that intersects $\Omega(L \cdot l)$ squares. □

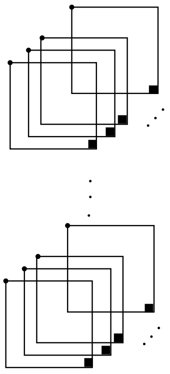

Figure 3: A vertical line intersects $\Omega(L \cdot l)$ squares