Geographic Quorum Systems Approximations

Paz Carmi* Shlomi Dolev† Sariel Har-Peled‡ Matthew J. Katz§
Michael Segal¶

November 30, 2003

Abstract

Quorum systems are a mechanism for obtaining fault-tolerance and efficient distributed systems. We consider *geographic quorum systems*; a geographic quorum system is a partitioning of a set $\mathcal{X}$ of points in the plane (representing servers) into quorums (i.e. clusters) of size k. The distance between a point p and a cluster $\mathcal{C}$ is the Euclidean distance between p and the farthest point in $\mathcal{C}$.

We present a near linear time constant-factor approximation algorithm for partitioning $\mathcal{X}$ into clusters, such that, the maximal distance between a point in the underlying region and its closest cluster is minimized. Next, we describe a data-structure for answering (approximately) nearest-neighbor queries on such a clustering.

Finally, we describe constant-factor approximation algorithms that associate regions of equal area to the servers in a given set of servers. Two cost measures are considered: the maximum distance of a point to its corresponding server, and the sum of average distances to the servers.

1 Introduction

A *quorum system* is an important abstraction used in distributed systems for achieving *fault-tolerance*, *availability* and *load balancing*, see [Her87, NW98, PW95, MR98, SP99, DGL+03].

*Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel, carmip@cs.bgu.ac.il. Partially supported by grant no. 2000160 from the U.S.-Israel Binational Science Foundation, and by a Kreitman Foundation doctoral fellowship.

†Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel, dolev@cs.bgu.ac.il. Partially supported by IBM faculty award, NSF grant, the Israeli ministry of defense, the Israeli Ministry of Industry and Trade (STRIMM consortium), and Rita Altura trust chair in computer sciences.

‡Department of Computer Science, DCL 2111; University of Illinois; 1304 West Springfield Ave.; Urbana, IL 61801; USA; http://www.uiuc.edu/~sariel/; sariel@uiuc.edu. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

§Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel, matya@cs.bgu.ac.il. Partially supported by grant no. 2000160 from the U.S.-Israel Binational Science Foundation, and by the MAGNET program of the Israel Ministry of Industry and Trade (LSRT consortium).

¶Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel, segal@cse.bgu.ac.il.

A quorum system is a collection of possibly intersecting subsets of servers, called here *quorums* or *clusters* ([Her87, DGL+03]). The *fault tolerance* or *availability* of a quorum system is related to the number of server failures that it can tolerate. E.g., there must remain at least one cluster without faulty servers, for each type of operation. The *load* of a quorum system is usually determined by the number of clusters a server belongs to. We suggest an additional new measure for quorum systems, namely the *communication cost* of accessing a cluster, which is an important aspect in the design of distributed quorum systems. In the context of mobile computing, the new communication cost measure raises several problems in computational geometry, that are of significant interest independent of the quorum system application.

The settings considered here are related to subdivisions of the plane [DGL+03], such that the cluster that is used by a wondering client is determined by the region in which the client is located. A *cluster* in our problems always consists of exactly k servers. We seek an optimal partition of the servers into clusters according to a certain cost measure. For example, the partition should imply low energy transmission of a mobile wondering ad hoc host, see [NI97, DSW02, DGL+03]. Roughly speaking we would like to have a (geographically) close enough cluster for every point in the plane.

Once a partition into clusters is defined, a mobile host can write a value to the servers of a near-by cluster, knowing that a later read operation must access a representative of each cluster and thus, access a server in the particular cluster that the mobile host used for the write operation. The question of whether to use a near-by cluster for writing and an intersecting set of servers for reading, or vice versa, may depend on the frequency of the write and read operations (see [DGL+03]).

Our results. In this paper we introduce several new problems concerning geographic quorum systems. Besides their obvious relevance to the study of quorum systems, these problems are also interesting as purely geometric problems. The first problem deals with the issue of constructing a good quorum system for a given set of servers and a service region R.

Geographic quorum partition. The task we address is to partition the set of n servers (represented by a set $\mathcal{X}$ of n points in the plane) into quorums (i.e., clusters), each of size k, such that, the cost associated with the partitioning is minimal. The cost $\mu(\mathcal{Q})$ associated with a partitioning $\mathcal{Q}$ is $\max_{p\in R} \mathbf{d}_{\mathcal{Q}}(p)$, where $\mathbf{d}_{\mathcal{Q}}(p)$ is the distance between p and the cluster of $\mathcal{Q}$ that is the closest to p, and the distance between p and a cluster $\mathcal{C}$ of $\mathcal{Q}$ is the *maximum* (Euclidean) distance between p and a point in $\mathcal{C}$ (see Figure 1).

We present a 3-approximation for this problem. More precisely, we construct a quorum system $\mathcal{Q}$ (i.e., a partitioning as above), in $O(n(k + \log^2 n))$ time, such that, for *any* point $p \in R$, we have $\mathbf{d}_{\mathcal{Q}}(p) \leq 3\mathbf{d}_{\mathcal{Q}'}(p)$, where $\mathcal{Q}'$ is any quorum partition of $\mathcal{X}$. In particular, we have $\mu(\mathcal{Q}) \leq 3\mu(\mathcal{Q}_{\text{opt}})$, where $\mathcal{Q}_{\text{opt}}$ is the optimal quorum system. If one is satisfied with a $(3+\varepsilon)$-approximation, the running time improves to $O\big(n \log^2 n + n/(k\varepsilon^3) \log^2(1/\varepsilon)\big)$.

To facilitate this, we present a fast implementation of the incremental greedy algorithm, which repeatedly picks the smallest disc that contains k input points, and continues with the remaining points. Our algorithm has near linear running time if one is satisfied with a $(1+\varepsilon)$-approximation of the smallest disc at each iteration, and might be of independent interest.

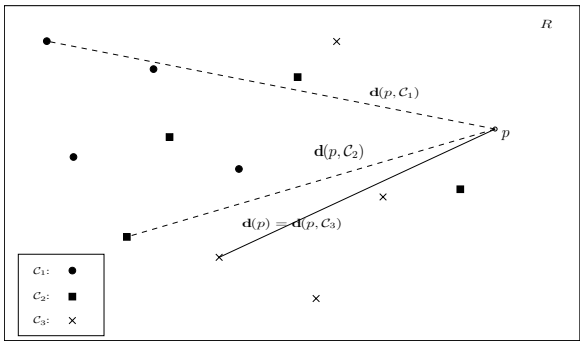

Figure 1: A partitioning into 3 clusters (c_1, c_2, c_3) of size 4, and the distance between point p and each of the clusters.

Nearest quorum queries. Given a quorum system $\mathcal{Q}$, we present a data-structure that answers queries of the form: which is the cluster that is closest (using the definition above) to the query location $q \in \mathbb{R}^2$. The natural data structure for such queries is the Voronoi diagram of the quorums; however, this diagram is too complex. We describe how to construct, in $O\big(n \log k + \frac{n}{k\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{\varepsilon}\big)$ time, a data structure of size $O(n/(k\varepsilon^2) \log(1/\varepsilon))$ that supports such queries in $O(\log(n/\varepsilon))$ time per query, where $\varepsilon > 0$ is a prespecified parameter. The distance between q and the cluster returned is at most $(1+\varepsilon)\mathbf{d}_{\mathcal{Q}}(q)$.

Load balancing. We suggest a complimentary research venue, in which the geographic quorum partition is given, and the goal is to balance the load among the clusters (and hence the servers).

Load balancing has been considered in the context of quorum systems; here however, we use geographic partitions and therefore, under the near-by quorum selection policy, the load may be distributed in an unbalanced manner. We note that the quorum partition that minimizes the cost may differ significantly from partitions that also attempt to balance the load. Moreover, when load balance is the concern it might be necessary at times to deviate from the near-by selection policy.

More precisely, under the near-by quorum selection policy, some of the quorums may be over utilized, while others may be under utilized. This happens when the region in which a quorum $\mathcal{C}$ is selected (i.e., its Voronoi cell) is either especially large or especially small (assuming uniform distribution of the customers in the region R.) It is therefore natural to consider the following problem. Divide the region R into m (where m is the number of clusters) connected subregions $R_1, \ldots, R_m$, each of area equal to area$(R)/m$, such that all requests initiated by customers in R_i are served by the ith cluster, and the cost of the division is minimal.

We study two versions of this problem, where each cluster $\mathcal{C}_i$ is represented by a single point p_i. In the first version (the min-max version), $\mu(p_i)$ is defined to be the maximum distance between p_i and a point in R_i, and the cost of the division is $max_i\, \mu(p_i)$. In the second version (the min-sum version), $\mu(p_i)$ is defined to be the average distance between p_i and a point in R_i, and the cost of the division is $\sum_i \mu(p_i)$. We are not aware of any previous results concerning this natural facility location problem. We present efficient constant-factor approximation algorithms for both versions. For example, for the min-max version we describe

an $O(n^{1.5} \log n)$ algorithm that computes a division that is a $(1 + \frac{3}{2}\sqrt{2\pi})$-approximation, i.e., a division such that the ratio between its cost and the cost of the corresponding optimal division is at most $1 + \frac{3}{2}\sqrt{2\pi}$, and for the min-sum version we describe an $O(n^{2+\epsilon})$ algorithm that computes a $(5 + 3\sqrt{2\pi})$-approximation. (If, instead of the region R, we are given a discrete set of customers, then both versions can be solved in time polynomial in m and the number of customers; the min-sum version is a special case of the *transportation* problem.)

The rest of the paper is organized as follows. The geographic quorum partition problem is studied in Section 2. A fast algorithm for computing the quorum partition is presented in Section 3. A data structure supporting approximate nearest quorum queries for a given partition is described in Section 4. Section 5 addresses the case in which load balancing is required for a given partition.

2 Geographic Quorum Partition

In this section we study the problem of partitioning the set of servers into clusters, each of size k, so as to minimize the cost of the partitioning. Let $\mathcal{X}$ be a set of points in the plane representing the servers. Let R be the region for which the servers are required to provide service. We assume R is a simple, closed and connected region (e.g., an axis-parallel rectangle).

The *distance* between a point p in R and a cluster $\mathcal{C} \subseteq \mathcal{X}$ is defined as the maximum (Euclidean) distance between p and a point of $\mathcal{C}$ (see Figure 1). That is

$$\mathbf{d}(p, \mathcal{C}) = \max_{c \in \mathcal{C}} \|pc\| ,$$

where $\|pc\|$ is the Euclidean distance between p and c.

Let n denote the number of points in $\mathcal{X}$ and assume $n = km$, where $k \geq 2$ and $m \geq 2$ are integers. Let $\mathcal{Q}$ be a partitioning of $\mathcal{X}$ into m clusters, each of size k. The *cost* of $\mathcal{Q}$, denoted by $\mu(\mathcal{Q})$, is a positive real number that is defined as follows. For each point $p \in R$, let $\mathbf{d}_{\mathcal{Q}}(p)$ be the distance between p and the cluster of $\mathcal{Q}$ that is the closest to p (according to the distance defined above). Then $\mu(\mathcal{Q}) = \max_{p \in R} \mathbf{d}_{\mathcal{Q}}(p)$.

We wish to compute a partitioning of $\mathcal{X}$ into k-clusters (i.e., clusters of size k) with minimum cost. We define a specific partitioning (called mec) of $\mathcal{X}$ into k-clusters, and prove that it is a 3-approximation; that is, its cost is at most three times the cost of an optimal partitioning, and it can be computed in polynomial time. Actually, we prove a stronger claim stating that for *any* point $q \in R$, $\mathbf{d}_{\mathsf{mec}}(q)$ is at most 3 times $\mathbf{d}_{\mathrm{opt}}(q)$ in opt, where opt is any partitioning of minimal cost.

We now define the partitioning mec. Among all subsets of $\mathcal{X}$ of size k, let $\mathcal{C}$ be the subset whose smallest enclosing circle is minimal. We take $\mathcal{C}$ as our first cluster and repeat for the set $\mathcal{X} \setminus \mathcal{C}$ of remaining points, until we are left with an empty set of points.

Lemma 2.1 *Let* opt *be a partitioning of $\mathcal{X}$ into k-clusters with minimum cost and let* mec *be the partitioning defined above. Then $\mu(\mathsf{mec}) \leq 3\mu(\mathrm{opt})$. Moreover, for any point $q \in R$, we have $\mathbf{d}_{\mathsf{mec}}(q) \leq 3\mathbf{d}_{\mathcal{Q}}(q)$, where $\mathcal{Q}$ is any quorum partition of $\mathcal{X}$.*

Proof: Let q be a point in R and let $\mathcal{A}$ be the cluster of mec that is the closest to q. That is, $\mathbf{d}_{\mathsf{mec}}(q) = \mathbf{d}(q, \mathcal{A})$. Let $\mathcal{C}$ be the cluster of opt that is the closest to q. Then we have $\mathbf{d}(q, \mathcal{C}) \geq r_{\mathcal{C}}$, where $r_{\mathcal{C}}$ is the radius of the smallest enclosing circle of $\mathcal{C}$.

Assume that $\mathcal{C}$ is not one of the clusters of mec (since otherwise $\mathbf{d}_{\mathsf{mec}}(q)$ is at most $\mathbf{d}_{\mathrm{opt}}(q)$ and we are done). Consider the (at most k) clusters of mec that have a non-empty intersection with $\mathcal{C}$. For at least one of them, $\mathcal{A}'$, we must have $r_{\mathcal{A}'} \leq r_{\mathcal{C}}$, since otherwise, C would have been created during the construction of mec. Let x be a point in $\mathcal{C} \cap \mathcal{A}'$. Let x' be the point in $\mathcal{A}'$ that is the farthest from q. Then

$$\mathbf{d}(q, \mathcal{A}) \leq \mathbf{d}(q, \mathcal{A}') = \|qx'\|,$$

and

$$\|qx'\| \leq \|qx\| + \|xx'\| \leq \mathbf{d}(q, \mathcal{C}) + 2r_{\mathcal{A}'},$$

and since $r_{\mathcal{A}'} \leq \mathbf{d}(q, \mathcal{C})$, we obtain that

$$\mathbf{d}(q, \mathcal{A}) \leq 3\mathbf{d}(q, \mathcal{C}).$$

In particular, if q is the point that determines $\mu(\mathsf{mec})$, then we have $\mu(\mathsf{mec}) = \mathbf{d}(q, \mathcal{A}) \leq 3\mathbf{d}(q, \mathcal{C}) \leq 3\mu(\mathrm{opt})$. ∎

The partitioning mec, for $k > 2$, can be computed by, e.g., repeatedly applying the recent algorithm of Har-Peled and Mazumdar for computing a smallest enclosing circle containing k points [HM03]. Their algorithm uses $O(n + k^2)$ space and its expected running time is $O(nk)$. For $k = 2$ the partitioning mec is computed by simply picking, at each iteration, the closest pair of points from the remaining points. This matching can be computed in $O(n \log n)$ time. The following theorem summarizes our results.

Theorem 2.2 *One can compute a partitioning $\mathcal{Q}$ of $\mathcal{X}$ into k-clusters, such that, for any point $q \in R$, we have $\mathbf{d}_{\mathcal{Q}}(q) \leq 3\mathbf{d}_{\mathrm{opt}}(q)$. For $k > 2$, $\mathcal{Q}$ can be computed in expected time $O(n^2)$ using $O(n + k^2)$ space. For $k = 2$, $\mathcal{Q}$ can be computed in time $O(n \log n)$ using linear space.*

3 A faster clustering algorithm

Let T be a compressed quadtree computed over a set $\mathcal{X}$ of n points in the plane, which can be computed in $O(n \log n)$ time [AMN+98]. We trim T by removing all nodes v, such that the parent of v has less than $c \cdot k$ points in its associated cell, where $c > 1$ is a constant to be specified shortly. A point $x \in \mathcal{X}$ is stored in a leaf v of T, if $x \in r_v$, where r_v is the cell associated with v.

Let us assume, that we are "lucky", and the smallest enclosing disc that contains k points is fully contained in one of the leaves of T. Clearly, every leaf v of T contains $|P_v| = O(k)$ points, and as such, we can in $O(|P_v|)$ time compute a constant factor approximation to the smallest disc that contains k points of P_v [HM03]. Applying this minimal disc computation to all the leaves of T, in advance, takes $O(n)$ time overall.

Let D_1 be the smallest disc that contains k points that is precomputed in one of the leaves of T. If we are lucky then D_1 is a constant factor approximation to the smallest disc

that contains k points of $\mathcal{X}$, and as such, we set the k points covered by D_1, denoted by $\mathcal{C}_1$, to be the first cluster.

We continue with the remaining points $\mathcal{X}_2 = \mathcal{X} \setminus \mathcal{C}_1$. To this end, we update T by deleting from it the points of $\mathcal{C}_1$, and performing trimming if needed, so that T becomes legal again.

Since all the points of $\mathcal{C}_1$ lie in a single leaf of T, all we need to do, is to climb up in T, till we reach a node of T that its parent has at least ck points in its cell. Since every time we go one level up in T, the cardinality of the set in the associated cell increases by at least one, it follows that we need to climb at most $O(k)$ levels to reach this node w (clearly, we need to trim all the children of w from T, as they contain too few points of $\mathcal{X}_2$ to be active).

To facilitate this, we will maintain an external data-structure of orthogonal range searching with deletions. Such that, at every node, we simply perform a range searching query to decide how many points are in the cell associated with this node. This takes $O(\log^2 n)$ per query and update [Aga97]. Thus, after $O(k \log^2 n)$ work, we found the node w, we trimmed its children, and we recomputed (approximately) the smallest disc that contains k of its points. The compressed quadtree T is now updated, and we can continue to the extraction of the next cluster.

If we are lucky again, and the smallest disc containing k points of $\mathcal{X}_2$ is completely contained in the cell of one of the leaves of T, then we can again extract a constant approximation to it as described above, and repeat. Thus, assuming (unreasonably) that we are lucky throughout the execution of the algorithm, we get a clustering which is constant factor competitive in $O(n \log^2 n)$ time.

Let us first consider the case where the current disc D_i (which is, say, an α-approximation to the smallest disc covering k points of $\mathcal{X}_i$) and the associated cluster $\mathcal{C}_i$ is given to us, and we would like to update T by removing the points of $\mathcal{C}_i$ from it.

Observe that D_i intersects at most a constant number of leaves of T. Indeed, let L be the set of leaves of T that intersect D_i. Since the parent of every node of L contains at least ck points of $\mathcal{X}_i$, it follows that its side length must be at least $\sqrt{2c}\,\mathrm{radius}(D_i)/\alpha$, otherwise D_i would not be a α-approximation to the smallest disc that covers k points of $\mathcal{X}_i$ (indeed, such a node v contains a disc of radius $\mathrm{SideLen}(r_v)/\sqrt{2c}$ that contains k points). Thus, let L' be the set of parents of nodes of L. Clearly, the regions that correspond to cells of nodes of L' are disjoint, and it follows that $|L'| = O(1)$ as they are all relatively large compared to D_i, and they all intersect D_i. Since every node of L' has a constant number of children, it follows that $|L| = O(1)$.

Thus, applying the algorithm described above for updating T to every leaf in L, implies that given $\mathcal{C}_i$ and D_i, we can update T to be the (trimmed) compressed quadtree of $\mathcal{X}_{i+1}$ in $O(k \log^2 n)$ time. Thus, we can maintain a trimmed quadtree throughout the clustering extraction process in $O(n \log^2 n)$ time.

We are still left with the task of guaranteeing that luck is on our side. A natural approach, would be to use several quadtrees, and to observe that it is sufficient to be lucky in one of them. In particular, one can generate such a set of compressed quadtrees by using random translations for the location of the starting bounding square of the quadtree. But in fact, no randomization is necessary, as a deterministic scheme is known. Indeed, assume that the point set is contained in the cube $C = [0, 1/\sqrt{d}]^d$ (this can be easily guaranteed by scaling the point set), and that the square used to compute the quadtree is $S = [-1, 1]^d$, then the

analysis of Chan [Cha98, Lemma 3.3] implies the following:

Lemma 3.1 *Assume d is even, and let $v^i = \left(\frac{i}{d+1}, \ldots, \frac{i}{d+1}\right)$ for $i = 0, \ldots, d$, $\mathcal{X}$ be a set of n points in $\mathbb{R}^d$ contained inside $C = [0, 1/\sqrt{d}]^d$, and $T_0, \ldots, T_d$ be the quadtrees of $\mathcal{X}$ having $v^0 + [-1, 1]^d, \ldots, v^d + [-1, 1]^d$ as the starting region for their root, respectively.*

Then, for any point $p \in C$, and $r \leq 1$, such that $\mathrm{ball}(p, r)$ contains a point of $\mathcal{X}$, there exists a cell R in one of those $d+1$ quadtrees, such that $\mathrm{ball}(p, r) \subseteq R$, and furthermore, the side length of R is bounded by $4(d+1)r$.

This lemma implies that we can maintain 3 quadtrees (since $d = 2$ in our settings), and we are guaranteed to be lucky in one of them. Indeed, if the disc D_i, of radius r_i, is an α-approximation to the smallest disc that contains k points of $\mathcal{X}_i$, then by a packing argument, the cell that contains D_i in its interior, contains at most $\lceil (4(d+1)r_i)^2/(r_i/\alpha)^2 \rceil\, k = O(k)$ points of $\mathcal{X}_i$ (otherwise, there would be a disc of radius smaller than r_i/α that contains k points of $\mathcal{X}_i$). Thus, setting $c = \lceil 16(d+1)^2\alpha^2 \rceil$, we are guaranteed to be lucky at one of the 3 quadtrees that we are maintaining. Thus, we just proved the following:

Lemma 3.2 *Given a set point $\mathcal{X}$ in the plane and a parameter k, one can compute, in $O(n \log^2 n)$ time, a constant factor competitive quorum clustering of $\mathcal{X}$.*

Computing exactly the smallest enclosing disc of k points, given $O(k)$ points, takes $O(k^2)$ time. Also, given an $\varepsilon > 0$, one can compute a $(1+\varepsilon)$-approximation to the smallest disc containing k points, when given $O(k)$ points, in $O\big(k + (1/\varepsilon^3)\log^2(1/\varepsilon)\big)$ time. Plugging this into the above algorithm we get the following:

Theorem 3.3 *Given a set point $\mathcal{X}$ in the plane and a parameters k and $\varepsilon > 0$, one can compute:*

1. *In $O(n(k + \log^2 n))$ time, a 3-competitive quorum clustering of $\mathcal{X}$.*
2. *In $O\big(n \log^2 n + \frac{n}{k\varepsilon^3} \log^2 \frac{1}{\varepsilon}\big)$ time, a $(3+\varepsilon)$-competitive quorum clustering of $\mathcal{X}$.*

Note, that the result of Theorem 3.3, can be further improved by using a better auxiliary data-structure for orthogonal range searching, or avoiding it all together. Since this only results in a minor improvement in the running time of the algorithm and substentially complicates the description of the algorithm, we did not persue this minor improvement.

4 Nearest Quorum Queries

Assume we already have a partitioning $\mathcal{Q}$ of a point set $\mathcal{X}$ into k-clusters (i.e., quorums). We now consider the following problem. Construct a data structure, so that given a query point $q \in \mathbb{R}^2$ (representing an ad-hoc client) one can quickly find the cluster (i.e., quorum) $\mathcal{C}^*$ of $\mathcal{Q}$ that is closest to q. (Recall that the distance $\mathbf{d}(q, \mathcal{C})$ between q and a cluster $\mathcal{C}$ is the maximum (Euclidean) distance between q and a point of $\mathcal{C}$.)

Given a prescribed parameter $\varepsilon > 0$, we show in this section, how to preprocess $\mathcal{Q}$, such that given any query point q in the plane, one can quickly decide what is (approximately) the nearest cluster of $\mathcal{Q}$ to q.

Assume that $\mathcal{Q}$ partitions $\mathcal{X}$ into the clusters $\mathcal{C}_1, \ldots, \mathcal{C}_m$, and recall that $\mathbf{d}_{\mathcal{Q}}(q) = \min_{\mathcal{C}_i \in \mathcal{Q}} \mathbf{d}(q, \mathcal{C}_i)$. Thus, the function $\mathbf{d}_{\mathcal{Q}}(q)$ can be interpreted as a minimization diagram of the maximization diagrams of the clusters. This min of max structure makes finding the nearest neighbor cumbersome, and our first task is to replace the maximization by minimization. Namely, we want to replace the furthest neighbor Voronoi diagram of each cluster by a "regular" minimization Voronoi diagram.

Definition 4.1 A pair $\mathcal{S} = (S, w)$ is a *weighted set of points* if $S = \{p_1, \ldots, p_m\}$ is a finite set of points in $\mathbb{R}^d$, and $w(\cdot)$ is a function assigning non-negative weights to the points of S. We define the distance of a point p from the point p_i to be $V_{(p_i, w(p_i))}(p) = \|pp_i\| + w(p_i)$. We define $V_{\mathcal{S}}(p) = \min_{i=1}^{m} V_{(p_i, w(p_i))}(p)$. The function $V_{\mathcal{S}}(p)$ induces a natural subdivision $\mathcal{V}_{\mathcal{S}}$ of $\mathbb{R}^d$ into cells, known as the (additive) *weighted Voronoi diagram* of $\mathcal{S}$, such that the ith cell is the locus of all points closest to p_i in this distance function. As is well known, in the planar case, $\mathcal{V}_{\mathcal{S}}$ has complexity $O(m)$, and it can be computed in $O(m \log m)$ time (see [For87]).

Lemma 4.2 ([Har99]) *Any furthest neighbor Voronoi diagram of points in $\mathbb{R}^d$, can be ε-approximated by a (nearest neighbor) weighted Voronoi diagram, having $O((1/\varepsilon^d) \log (1/\varepsilon))$ sites.*

We shortly outline how to compute this approximation, for a point set U, and its furthest neighbor Voronoi diagram $\mathcal{F}_U$: (i) Find a point x for which the value of the furthest Voronoi diagram of U is minimal. This can be easily done by computing the smallest disc that contains U. (ii) Let $l = \mathcal{F}_U(x)$ be the distance of the furthest neighbor of x in U. Given l and x, the locations of the approximation sites are now determined, and let S be this set of $O(1/\varepsilon^2 \log 1/\varepsilon)$ sites. Next, we compute the furthest neighbor Voronoi diagram of U, and for each point s of S, we compute its weight $\mathcal{F}_U(s)$, by performing a point-location query in the furthest neighbor Voronoi diagram of $\mathcal{F}_U$ (i.e., $w(s)$ is set to $\mathcal{F}_U(s)$). The resulting weighted additive Voronoi diagram (S, w) is the required approximation. We summarize this in the following lemma:

Lemma 4.3 *Given a cluster C of k points in the plane and a parameter $\varepsilon > 0$, one can compute, in $O(k \log k + (\log k)/\varepsilon^2 \log 1/\varepsilon)$ time, an additive weighted Voronoi diagram $\mathcal{V}_C$, with $O(1/\varepsilon^2 \log(1/\varepsilon))$ sites, such that $\mathcal{V}_C$ is a $(1+\varepsilon)$ approximation to the furthest neighbor Voronoi diagram $\mathcal{F}_C$. Formally, for any point $x \in \mathbb{R}^2$, we have $\mathcal{F}_C(x) \leq \mathcal{V}_C(x) \leq (1+\varepsilon)\mathcal{F}_C(x)$.*

Let us apply Lemma 4.3 to each cluster $\mathcal{C}_i$ in $\mathcal{Q}$, and let $\mathcal{V}_i$ be the resulting weighted additive Voronoi diagram and $\mathcal{S}_i$ be the weighted set that induces it, for $i = 1, \ldots, m$. Clearly, an approximate nearest neighbor query $q \in \mathbb{R}^2$ in the quorum clustering $\mathcal{Q}$, can be resolved by finding the i that realizes

$$\min_i \mathcal{V}_i(q) = \min_i \min_{(s,w) \in \mathcal{S}_i} (\|sq\| + w) = \min_{(s,w) \in \bigcup_i S_i} (\|sq\| + w),$$

but this is exactly the value of the weighted additive Voronoi diagram of $\mathcal{S} = \cup_i \mathcal{S}_i$ at the point q. Thus, if we precompute the diagram $\mathcal{V}_{\mathcal{S}}$, then answering approximate nearest neighbor

queries on $\mathcal{Q}$, is equivalent to performing a point-location query in $\mathcal{V}_{\mathcal{S}}$. We summarize the result:

Theorem 4.4 *Let $\varepsilon > 0$ be a parameter, and $\mathcal{Q}$ be a given quorum clustering of a set $\mathcal{X}$ of n points in the plane, where each cluster is of cardinality k. One can preprocess $\mathcal{Q}$ in $O\big(n \log k + \frac{n}{k\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{\varepsilon}\big)$ time and $O(n/(k\varepsilon^2) \log(1/\varepsilon))$ space, such that given a query point q, one can decide in $O(\log(n/\varepsilon))$ time what is the $(1+\varepsilon)$-approximate nearest neighbor cluster to q among the clusters of $\mathcal{Q}$.*

Proof: We just verify the time and space bounds. Computing the approximation to each cluster takes $O(k \log k + (\log k)/\varepsilon^2 \log 1/\varepsilon)$ time. Once this approximation is computed, we have to compute the additive Voronoi diagram of all the approximations together. Since there are $O(n/(k\varepsilon^2) \log(1/\varepsilon))$ sites, this takes $O\big(n \log k + \frac{n}{k\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{k\varepsilon}\big)$ time. Thus, the overall running time is

$$O\left(\frac{n}{k}\left(k \log k + \frac{\log k}{\varepsilon^2} \log \frac{1}{\varepsilon}\right) + \frac{n}{k\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{k\varepsilon}\right) = O\left(n \log k + \frac{n}{k\varepsilon^2} \log \frac{1}{\varepsilon} \log \frac{n}{\varepsilon}\right).$$

As for the space, we build a point-location data structure for $O(n/(k\varepsilon^2) \log(1/\varepsilon))$ sites, and this is the amount of space needed for such a data structure. ∎

5 Load Balancing

In this section we consider a different aspect of geographic quorum systems, namely balancing the load among clusters. Given a partitioning into k-clusters, one would like to divide the load among the clusters, while keeping the total cost low. In other words we wish to divide the region R into m connected subregions $R_1, \ldots, R_m$, each of area $\text{area}(R)/m$, such that all requests initiated by clients in R_i are served by the ith cluster, and the total cost is minimal.

Let us re-formulate the problem more precisely. We assume that the number m of clusters is equal to l^2, for some integer $l \geq 2$, and that each quorum $\mathcal{C}_i$ is represented by a single point p_i (e.g., the center of the smallest enclosing circle of $\mathcal{C}_i$). Put $\mathcal{P} = \{p_1, \ldots, p_m\}$. We also assume (for convenience only) that the underlying region R is a square. (This assumption is not necessary; all subsequent results hold for any rectangle R that can be divided into m squares of equal size. Thus the aspect ratio of R can be as large as m.) We study the following problem. Divide R into m connected regions $R_1, \ldots, R_m$, each of area $\text{area}(R)/m$, such that region R_i is associated with point p_i, and the total cost is minimal.

We consider two cost measures. In the first measure, $\mu(p_i)$ is defined to be the maximum distance between p_i and a point in R_i, and the cost of the division is $max_i\, \mu(p_i)$. In the second measure, $\mu(p_i)$ is defined to be the average distance between p_i and a point in R_i, and the cost of the division is $\sum_i \mu(p_i)$.

For each of the two versions, we describe a simple algorithm that divides R into m subregions and associates them with the representative points in $\mathcal{P}$. We call the division obtained for the min-max version (using the first cost measure) GRID-MIN-MAX, and the division obtained for the min-sum version (using the second cost measure) GRID-MIN-SUM. We then prove that GRID-MIN-MAX is a $(1 + \sqrt{2\pi})$-approximation, meaning that the ratio

between the cost of GRID-MIN-MAX and the cost of an optimal division for the min-max version is at most $(1+\sqrt{2\pi})$. And that GRID-MIN-SUM is a $(5+\frac{3}{2}\sqrt{2\pi})$-approximation, under the assumption that the subregions must be convex. The running time of our algorithm for the min-max versions is $O(n^{2.5}\sqrt{\log n})$, and the running time for the min-sum version is $O(n^3)$.

Both algorithms divide R into $m = l^2$ squares of equal size, and associate the squares with the points in $\mathcal{P}$. Let $\mathcal{S}$ denote the set of squares. Let $G = (\mathcal{P}, \mathcal{S}; E)$ be the complete bipartite graph with vertex sets $\mathcal{P}$ and $\mathcal{S}$. We now associate weights with the edges in E. In the min-max version, the weight of the edge (p, σ), for $p \in \mathcal{P}$ and $\sigma \in \mathcal{S}$, is the maximum distance between p and a point in σ. This distance is clearly determined by one of the corners of σ. In the min-sum version, the weight of the edge (p, σ) is the average distance between p and the points in σ. In the min-max version we associate the squares in $\mathcal{S}$ with the points in $\mathcal{P}$ by computing a bottleneck matching in G, i.e., a matching in which the weight of the heaviest edge is minimal. Using the algorithm of Gabow and Tarjan et al. [GT88] this can be done in $O(n^{2.5}\sqrt{\log n})$ time. In the min-sum version we associate the squares with the points by computing a minimum-weight matching in G, i.e., a matching for which the sum of the weights (of the m edges defining the matching) is minimal. Using the algorithm of Kuhn [Kuh95] this can be done in $O(n^3)$ time.

We now prove that the divisions that were obtained (i.e., GRID-MIN-MAX and GRID-MIN-SUM) are constant-factor approximations.

5.1 GRID-MIN-MAX is a constant-factor approximation

Let opt1 denote an optimal division for the min-max version (where region R_i is associated with point p_i). We use opt1 to obtain a new division, grid1, that is also based on the squares in $\mathcal{S}$. We then show that grid1 is a $(1 + \sqrt{2\pi})$-approximation, immediately implying that GRID-MIN-MAX is also a $(1 + \sqrt{2\pi})$-approximation, since GRID-MIN-MAX is the best division among those based on the squares in $\mathcal{S}$.

Define a bipartite graph $G = (\mathcal{S}, \mathcal{P}; E)$, where there is an edge between $\sigma_i \in \mathcal{S}$ and $p_j \in \mathcal{P}$ if $\sigma_i \cap R_j \neq \emptyset$. Hall's matching theorem [Wes01] gives a necessary and sufficient condition for G to contain a perfect matching. According to Hall's theorem, G contains a perfect matching if and only if for any subset $\mathcal{S}'$ of $\mathcal{S}$ we have $|N(\mathcal{S}')| \geq |\mathcal{S}'|$, where $N(\mathcal{S}')$ is the set of points in $\mathcal{P}$ that are connected by an edge to a square in $\mathcal{S}'$. However, this condition trivially holds in our case, since we need at least $|\mathcal{S}'|$ regions of $\mathcal{R}$ in order to cover a region of area $|\mathcal{S}'|\text{area}(R)/m$. We thus associate the squares in $\mathcal{S}$ with the points in $\mathcal{P}$ to obtain the division grid1 by computing any perfect matching in G.

Let $p_i \in \mathcal{P}$ and let $\sigma_j \in \mathcal{S}$ be the square that was assigned to p_i by grid1. Recall that the region assigned to p_i by opt1 is R_i. Let $q \in \sigma_j \cap R_i$. Assume w.l.o.g. that $\text{area}(\sigma_j) = \text{area}(R_i) = 1$, then the diameter of R_i is at least $2/\sqrt{\pi}$ (see Figure 2). Therefore, $\mu_{\text{opt1}}(p_i)$ is, on the one hand, at least $1/\sqrt{\pi}$, and, on the other hand, at least $\|p_iq\|$. As to $\mu_{\text{grid1}}(p_i)$ we have $\mu(p_i) \leq \|p_iq\| + \sqrt{2}$. Now, if $\|p_iq\| \leq 1/\sqrt{\pi}$, then using the first inequality for $\mu_{\text{opt1}}(p_i)$ we obtain that

$$\frac{\mu_{\text{grid1}}(p_i)}{\mu_{\text{opt1}}(p_i)} \leq \frac{\|p_iq\| + \sqrt{2}}{1/\sqrt{\pi}} \leq 1 + \sqrt{2\pi},$$

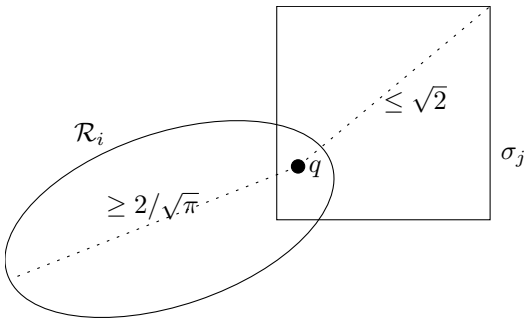

Figure 2: grid1 (alternatively grid2) is a constant-factor approximation.

and, if $\|p_i q\| > 1/\sqrt{\pi}$, then using the second inequality for $\mu_{\mathsf{opt1}}(p_i)$ we obtain that

$$\frac{\mu_{\mathsf{grid1}}(p_i)}{\mu_{\mathsf{opt1}}(p_i)} \leq \frac{\|p_i q\| + \sqrt{2}}{\|p_i q\|} \leq 1 + \sqrt{2\pi}.$$

We conclude that in both cases the ratio $\mu_{\mathsf{grid1}}(p_i)/\mu_{\mathsf{opt1}}(p_i) \leq 1 + \sqrt{2\pi}$, for any $1 \leq i \leq m$, and therefore grid1 is a $(1 + \sqrt{2\pi})$-approximation. (Let $p_k \in \mathcal{P}$ be the point that determines the cost of grid1. Then $\mu(\mathsf{grid1})/\mu(\mathsf{opt1}) \leq \mu_{\mathsf{grid1}}(p_k)/\mu_{\mathsf{opt1}}(p_k)$.) Finally, since the cost of GRID-MIN-MAX is at most $\mu(\mathsf{grid1})$, we conclude that GRID-MIN-MAX is a $(1 + \sqrt{2\pi})$-approximation.

Notice that if we restrict the regions of opt1 to be axis-parallel rectangles, then the diameter of R_i is at least $\sqrt{2}$, and GRID-MIN-MAX becomes a 3-approximation.

Theorem 5.1 *A division of R that is a $(1 + \sqrt{2\pi})$-approximation can be computed in $O(n^{2.5}\sqrt{\log n})$ time.*

Consider the complete bipartite graph $G = (\mathcal{P}, \mathcal{S}; E)$ in which we compute a bottleneck matching to obtain the division GRID-MIN-MAX. By modifying the definition of the weight of an edge $(p, \sigma) \in E$, we are able to reduce the running time of our algorithm to $O(n^{1.5} \log n)$, without increasing the approximation factor too much. We define the weight of (p, σ) to be the distance between p and the center of σ. Now, the graph G is actually the complete bipartite graph induced by two point sets in the plane, and we can apply to it the algorithm of Efrat et al. [EIK01] that computes a bottleneck matching in such graphs in time $O(n^{1.5} \log n)$.

It remains to bound the approximation factor of the division GRID-MIN-MAX' that is obtained. Let r (resp. r') be the weight of the heaviest edge in the bottleneck matching defining GRID-MIN-MAX (resp. GRID-MIN-MAX'). Clearly, $r' \leq r$. The cost of GRID-MIN-MAX' is less than or equal to the cost of GRID-MIN-MAX plus $\sqrt{2}/2$, since the cost of GRID-MIN-MAX' is at most $r' + \sqrt{2}/2$ (and the cost of GRID-MIN-MAX is r). Therefore, using the fact that the cost of opt1 is at least $1/\sqrt{\pi}$, we obtain that GRID-MIN-MAX' is a $(1 + \frac{3}{2}\sqrt{2\pi})$-approximation.

Corollary 5.2 *A division of R that is $(1+\frac{3}{2}\sqrt{2\pi})$-approximation can be computed in $O(n^{1.5} \log n)$ time.*

5.2 GRID-MIN-SUM is a constant-factor approximation

Let opt2 denote an optimal division (into convex subregions) for the min-sum version (where region R_i is associated with point p_i). We use opt2 to obtain a new division, grid2, that is also based on the squares in $\mathcal{S}$. This is done, as in Section 5.1, by defining a bipartite graph $G = (\mathcal{S}, \mathcal{P}; E)$, where there is an edge between σ_i and p_j if $\sigma_i \cap R_j \neq \emptyset$, and computing a perfect matching in G. We now show that grid2 is a $(5+2\sqrt{2\pi})$-approximation, immediately implying that GRID-MIN-SUM is also a $(5+2\sqrt{2\pi})$-approximation, since GRID-MIN-SUM is the best division among those based on the squares in $\mathcal{S}$.

Let $p_i \in \mathcal{P}$ and let $\sigma_j \in \mathcal{S}$ be the square that was assigned to p_i by grid2. Let $q \in \sigma_j \cap R_i$, where R_i is the region assigned to p_i by opt2. Assume w.l.o.g. that $\text{area}(\sigma_j) = \text{area}(R_i) = 1$, then the diameter of R_i is at least $2/\sqrt{\pi}$. Let z_i be a point in the plane for which the average distance to the points in R_i is minimal. Since R_i is convex, z_i is the center of mass (centroid) of R_i, and $d_{avg}(z_i, R_i) \geq \Delta(R_i)/4$ and also $d_{avg}(z_i, R_i) \geq 2/(3\sqrt{\pi})$, where $\Delta(R_i)$ is the diameter of R_i. (The right side of the former inequality is equal to the average distance between the middle point of a segment of length $\Delta(R_i)$ and the points of the segment, and the right side of the latter inequality is equal to the average distance between the center point of a disc of radius 1 and the points of the disc.) Let $a_i \in R_i$ be the closest point to p_i (if $p_i \in R_i$, then $a_i = p_i$). Then $\mu_{\text{opt2}}(p_i)$ is, on the one hand, at least $d_{avg}(z_i, R_i)$, and, on the other hand, at least $\|p_i a_i\|$. As to $\mu_{\text{grid2}}(p_i)$ we have $\mu(p_i) \leq \|p_i a_i\| + \|a_i q\| + \sqrt{2} \leq \|p_i a_i\| + \Delta(R_i) + \sqrt{2}$.

Now, if $\|p_i a_i\| \geq d_{avg}(z_i, R_i)$, then using the second inequality for $\mu_{\text{opt2}}(p_i)$ (and noticing that in this case $\Delta(R_i) \leq 4\|p_i a_i\|$) we obtain that

$$\frac{\mu_{\text{grid2}}(p_i)}{\mu_{\text{opt2}}(p_i)} \leq \frac{\|p_i a_i\| + \Delta(R_i) + \sqrt{2}}{\|p_i a_i\|} \leq \frac{5\|p_i a_i\| + \sqrt{2}}{\|p_i a_i\|} \leq 5 + \frac{3}{2}\sqrt{2\pi},$$

and, if $\|p_i a_i\| < d_{avg}(z_i, R_i)$, then using the first inequality for $\mu_{\text{opt2}}(p_i)$ we obtain that

$$\frac{\mu_{\text{grid2}}(p_i)}{\mu_{\text{opt2}}(p_i)} \leq \frac{\|p_i a_i\| + \Delta(R_i) + \sqrt{2}}{d_{avg}(z_i, R_i)} \leq \frac{5 d_{avg}(z_i, R_i) + \sqrt{2}}{d_{avg}(z_i, R_i)} \leq 5 + \frac{3}{2}\sqrt{2\pi}.$$

We conclude that in both cases the ratio $\mu_{\text{grid2}}(p_i)/\mu_{\text{opt2}}(p_i) \leq 5 + \frac{3}{2}\sqrt{2\pi}$, for any $1 \leq i \leq m$, and therefore grid2 is a $(5+\frac{3}{2}\sqrt{2\pi})$-approximation. Finally, since the cost of GRID-MIN-SUM is at most $\mu(\text{grid2})$, we conclude that GRID-MIN-SUM is a $(5+\frac{3}{2}\sqrt{2\pi})$-approximation.

Theorem 5.3 *A division of R that is a $(5+\frac{3}{2}\sqrt{2\pi})$-approximation can be computed in $O(n^3)$ time.*

Consider the complete bipartite graph $G = (\mathcal{P}, \mathcal{S}; E)$ in which we compute a minimum weight matching to obtain the division GRID-MIN-SUM. By modifying the definition of the weight of an edge $(p, \sigma) \in E$, we can both simplify the computation of the edge weights and reduce the running time of our algorithm to $O(n^{2+\epsilon})$, without increasing the approximation factor too much. We define the weight of (p, σ) to be the distance between p and the center of σ. Now, the graph G is actually the complete bipartite graph induced by two point sets in the plane, and we can apply to it the algorithm of Agarwal et al. [AES99] that computes a minimum weight matching in such graphs in time $O(n^{2+\epsilon})$ using $O(n^{1+\epsilon})$ space.

It remains to bound the approximation factor of the division GRID-MIN-SUM' that is obtained. We show that the cost of GRID-MIN-SUM' is at most the cost of GRID-MIN-SUM plus $m\sqrt{2}$, and therefore GRID-MIN-SUM' is a $(5+3\sqrt{2\pi})$-approximation (using the inequality $\mu(\mathsf{opt2}) \geq 2m/(3\sqrt{\pi})$). Indeed, let M (resp., M') be the matching defining GRID-MIN-SUM (resp., GRID-MIN-SUM'). Also, for a point p_i and a square σ_j, let $q_j^i \in \sigma_j$ be the closest point to p_i, and let o_j be the center of σ_j. Then

$$m\sqrt{2} + \sum_{(p_i,\sigma_j)\in M} d_{avg}(p_i,\sigma_j) \geq \frac{m\sqrt{2}}{2} + \sum_{(p_i,\sigma_j)\in M} (\|p_i q_j^i\| + \frac{\sqrt{2}}{2}) \geq$$

$$\frac{m\sqrt{2}}{2} + \sum_{(p_i,\sigma_j)\in M} \|p_i o_j\| \geq \frac{m\sqrt{2}}{2} + \sum_{(p_i,\sigma_j\in M'} \|p_i o_j\| = \sum_{(p_i,\sigma_j)\in M'} (\|p_i o_j\| + \frac{\sqrt{2}}{2}).$$

But the first expression in the above sequence of inequalities is the cost of GRID-MIN-SUM plus $m\sqrt{2}$, and the last expression is greater or equal than the cost of GRID-MIN-SUM'.

Corollary 5.4 *A division of R that is $(5+3\sqrt{2\pi})$-approximation can be computed in $O(n^{2+\epsilon})$ time.*

References

[AES99] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29:912–953, 1999.

[Aga97] P. K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 31, pages 575–598. CRC Press LLC, Boca Raton, FL, 1997.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. Assoc. Comput. Mach.*, 45(6), 1998.

[Cha98] T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comput. Geom.*, 20:359–373, 1998.

[DGL+03] S. Dolev, S. Gilbert, N. A. Lynch, A. Shvartsman, and J. L. Welch. Geoquorums: Implementing atomic memory in ad hoc networks. In *Proc. of the 17th Int. Symp. Dist. Comp.*, 2003.

[DSW02] S. Dolev, E. Schiller, and J. Welch. Walk for self-stabilizing group communication in ad-hoc networks. In *Proc. 21st IEEE Symp. Reliable Dist. Sys.*, pages 70–79, 2002.

[EIK01] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[For87] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[GT88] H. N. Gabow and R. E. Tarjan. Algorithms for two bottleneck optimization problems. *J. Algorithms*, 9:411–417, 1988.

[Har99] S. Har-Peled. Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput.*, 28(4):1182–1197, 1999.

[Her87] M. P. Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Transactions on Database Systems*, 12(2):170–194, June 1987.

[HM03] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing disc. In *Proc. 11th Annu. European Sympos. Algorithms*, volume 2832 of *Lect. Notes in Comp. Sci.*, pages 278–288. Springer-Verlag, 2003.

[Kuh95] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–98, 1995.

[MR98] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[NI97] J. C. Navas and T. Imielinski. GeoCast – geographic addressing and routing. In *Mobile Computing and Networking*, pages 66–76, 1997.

[NW98] M. Naor and A. Wool. The load, capacity, and availability of quorum systems. *SIAM J. Comput.*, 27(2):423–447, 1998.

[PW95] D. Peleg and A. Wool. The availability of quorum systems. *Information and Computation*, 123(2):210–223, 1995.

[SP99] I. Stojmenovic and P. E. V. Pena. A scalable quorum based location update scheme for routing in ad hoc wireless networks. Technical report, Computer Science, SITE, University of Ottawa, December 1999. TR-99-11.

[Wes01] D. B. West. *Intorudction to Graph Theory*. Prentice Hall, 2ed edition, 2001.

A Some observations

Lemma A.1 *Let P be a convex polygon in the plane, and let D be a disc of radius $\Delta(P)/6$, where $\Delta(P)$ denote the diameter of P. Then,* $\text{area}(P \setminus D) \geq \text{area}(D)/9$.

Proof: Let p, q be the diameterical pair of points of P, and let $Q = f(P)$ be the polygon resulting from P by shrinking it by a factor of 3, by the transformation $f(x, y) = (x/3, y/3)$. Place a copy Q_1 of Q such that it is contained inside P, and it has a common tangent with P at the point p, and similarly, place a copy $Q_2 \subseteq P$ of Q such that it has a common tangent with P at q.

Clearly, the distance between any pair of points $x \in Q_1$ and $y \in Q_2$ is at least $\Delta(P)/3$. In particular, D can not intersect both Q_1 and Q_2. Assume that it does not intersect Q_1. Thus, we have $\text{area}(P \setminus D) \geq \text{area}(Q_1) \geq \text{area}(P)/9$, as claimed. ∎

For a convex polygon P and $y \in P$, let $\mu_P(y) = \int_{x \in P} \|xy\|$. Let $\mathcal{FW}_P = \arg\min_{y \in P} \mu_P(y)$ denote the *Fermat-Weber* center of P. By Lemma A.1, we have that $\mu_{\text{opt}}(P) = \mu_P(\mathcal{FW}_P) \geq \Delta(P)/54$. This implies that approximating $\mu_{\text{opt}}(P)$ is quite easy. Indeed, spread a grid G with side length $c\varepsilon^2\Delta(P)$, where c is an appropriate small constant. Let U be the set of all grid points that are inside P. Next, associate with each point p of U, the region r_p of P closest to it, and associate the weight $w_p = \text{area}(r_p)$ with p. For a point $y \in P$, let $\mu_U(y) = \sum_{p \in U} \|py\| w_p$. It is now straightfoward to verify the following.

Lemma A.2 *For any point $y \in P$, we have $(1-\varepsilon)\mu_P(y) \leq \mu_U(y) \leq (1+\varepsilon)\mu_P(y)$.*

In particular, finding the point $z \in U$ that minimizes $\mu_U(\cdot)$ results in an $(1+\varepsilon)$-approximation to $\mu_{\text{opt}}(P)$. Since there are $O(1/\varepsilon^4)$ points in U, it follows:

Lemma A.3 *Given a polygon P with n vertices, and a parameter ε, one can compute in $O(n + 1/\varepsilon^8)$ time a point $p \in P$, such that $\mu_P(p) \leq (1+\varepsilon)\mu_{\text{opt}}(P)$.*

The dependency on ε in Lemma A.3 can be improved. Since this is not crucial to our discussion, we omit further detials.