Incremental Codes

Yevgeniy Dodis* Shai Halevi†

Abstract

We introduce the notion of *incremental* codes. Unlike a regular code of a given rate, which is an unordered set of elements with a large minimum distance, an incremental code is an ordered vector of elements each of whose prefixes is a good regular code (of the corresponding rate). Additionally, while the quality of a regular code is measured by its minimum distance, we measure the quality of an incremental code $\mathcal{C}$ by its *competitive ratio* A: the minimum distance of *each* prefix of $\mathcal{C}$ has to be at most a factor of A smaller than the minimum distance of the *best* regular code of the same rate.

We first consider incremental codes over an arbitrary compact metric space M, and construct a 2-competitive code for M. When M is finite, the construction takes time $O(|M|^2)$, exhausts the entire space, and is NP-hard to improve in general. We also show optimal incremental codes for important specific spaces: the real interval $[0,1]$ and, most significantly, the Hamming space F^n over moderate alphabets ($|F| \geq n$). Finally, we concentrate our attention on Hamming spaces F^n over small alphabets. Obtaining good competitive ratio is somewhat hard in this case since our current knowledge of coding theory does not even yield very good regular codes of a given rate. Nevertheless, we give three efficient constructions of incremental codes over F^n achieving constant competitive ratios for various important settings of parameters.

*Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: dodis@cs.nyu.edu

†IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598, USA. Email: shaih@watson.ibm.com.

1 Motivating Example

Imagine the following problem which was actually given to one of the authors. An Internet company wants to assign account numbers to its customers when the latter shop on-line. An account number allows the customer to check the status of the order, get customer support, etc. In particular, the customer can enter it over the phone. Because of that and several other reasons, account numbers should not be too long. On the other hand, we would like account numbers to be somewhat far from each other, so that it is unlikely for the customer to access a valid number by mis-entering few digits. One way to achieve this would be to use an error-correcting code of reasonable minimum distance (for example, a random account number might work for a while). This has two problems, however. First, good distance implies not very good rate, and since the account numbers are quite short, we "waste" a lot possible account numbers, and exhaust our small account space too quickly (thus, losing customers). Secondly, when the number of customers is small, the corresponding prefix of our code is not as good as we could have made it with so few account numbers.

We propose a much better solution to this problem, namely, to use an *incremental* code. Such a code will eventually exhaust (or nearly exhaust) the whole space. Indeed, when the number of customers is huge, we prefer to have close accounts numbers rather than to lose customers. On the other hand, when the number of customers is i, incremental code *guarantees* that the first i account numbers we assign will be almost as far from each other as *any* possible i account numbers could be! In other words, the minimal distance of larger and larger prefixes of the code slowly decreases at an almost optimal pace.

Notice that while it is customary to measure regular codes of a given rate in terms of their minimum distance, a more relevant measure of incremental codes is the relative behavior of minimal distance on larger and larger prefixes. This leads to the notion of a *competitive ratio* of an incremental code $\mathcal{C}$. Namely, $\mathcal{C}$ is A-competitive if the minimum distance of each prefix of $\mathcal{C}$ is at most A times smaller than that of the *optimum* code of the same rate.

Organization. While the main motivation for incremental codes comes from the Hamming spaces, we start in Section 2 by defining and studying the corresponding notion on arbitrary (finite or even compact) metric spaces. In particular, we obtain a 2-competitive codes for any such metric space, and show that it is NP-hard (in general) to beat this competitive ratio. In Section 3 we give two optimal constructions for specific important spaces. One constructs an optimal incremental code for the real interval $[0, 1]$, while the other gives a simple and efficient 1-competitive code for the Hamming space over moderate alphabets. In particular, it gives an optimal and very practical solution to the "account problem" defined above. Finally, in Section 4 we concentrate in more detail on the intricacies of the Hamming space over small alphabets. While it is much harder to obtain competitive codes in this case (since our understanding of optimal codes is somewhat limited), we give several efficient constructions achieving constant competitive ratios.

2 General Notion and Construction

Here we formally define the concept of incremental codes and give their constructions for general (compact) metric spaces. To avoid verbosity, we first talk about arbitrary *finite* metrics, and later extend our results to any (possibly infinite) *compact* metrics.

So let $\mathcal{M} = (M, D)$ be any finite metric space on point set M with metric D. A (regular) *code* on $\mathcal{M}$ is simply a subset of points $S \subseteq M$. The minimum distance $d_{\mathcal{M}}(S)$ of S is the smallest pairwise distance between distinct points in S. For an integer i we define the quantity $\mathsf{opt}\text{-}d_{\mathcal{M}}(i)$ to be the largest

minimal distance of a code of cardinality i: $\mathsf{opt}\text{-}d_{\mathcal{M}}(i) = \max_{|S|=i} d_{\mathcal{M}}(S)$.

An *incremental code* $\mathcal{C} = \langle c_1 \dots c_k \rangle$ is an *ordered* sequence of distinct points of M. $\mathcal{C}$ is *exhaustive* if $k = |M|$, i.e. the code eventually runs through the entire space. For every $i \in [k]$ we define the i-th prefix of $\mathcal{C}$, $\mathcal{C}_i = \{c_1 \dots c_i\}$, and view it as a regular (unordered) code of cardinality i. We say that $\mathcal{C}$ is A-competitive, if for every $i \in [k]$, the i-th prefix $\mathcal{C}_i$ of $\mathcal{C}$ forms a code of distance at least $\mathsf{opt}\text{-}d_{\mathcal{M}}(i)/A$, i.e. $\mathsf{opt}\text{-}d_{\mathcal{M}}(i) \leq A \cdot d_{\mathcal{M}}(\mathcal{C}_i)$. We denote by $r_{\mathcal{M}}(\mathcal{C})$ the (best) competitive ratio of $\mathcal{C}$, and by $\mathsf{opt}\text{-}r_{\mathcal{M}}(k)$ the smallest competitive ratio of any incremental code of cardinality k: $\mathsf{opt}\text{-}r_{\mathcal{M}}(k) = \min_{|\mathcal{C}|=k} r_{\mathcal{M}}(\mathcal{C})$. We define $\mathsf{opt}\text{-}r_{\mathcal{M}} = \mathsf{opt}\text{-}r_{\mathcal{M}}(|M|)$, and call it the *competitive ratio* of $\mathcal{M}$. (We notice that since the prefix an A-competitive incremental code is also A-competitive, we have that $\mathsf{opt}\text{-}r_{\mathcal{M}}(k)$ is a non-decreasing function of k.) We say that an incremental code $\mathcal{C}$ is *perfect* if $\mathcal{C}$ is 1-competitive, and that the space $\mathcal{M}$ is *incrementally perfect* if it has an exhaustive 1-competitive code ($\mathsf{opt}\text{-}r_{\mathcal{M}} = 1$).

Theorem 1

1. *The competitive ratio of any $\mathcal{M}$ is at most 2:* $\mathsf{opt}\text{-}r_{\mathcal{M}} \leq 2$. *Moreover, given $\mathcal{M}$ as an input, one can construct an exhaustive 2-competitive incremental code $\mathcal{C}$ for $\mathcal{M}$ in time $O(|M|^2)$. In fact, constructing k-prefix of $\mathcal{C}$ can be done in time $O(k \cdot |M|)$.*
2. *There exist $\mathcal{M}$ with competitive ratio 2.*
3. *For any $A < 2$ and given $\mathcal{M}$ as an input, it is NP-hard to construct A-competitive incremental code for $\mathcal{M}$, even when the competitive ratio of $\mathcal{M}$ is known to be 1. In particular, it is NP-hard to approximate to competitive ratio of $\mathcal{M}$ to within a factor less than 2.*

Proof: Given a point p and a finite set of points S, define the distance from p to S to be $D(p, S) = \min_{q \in S} D(p, q)$. We use the following simple greedy algorithm for constructing $\mathcal{C}$.

1. Let c_1 be any point of M, and let $\mathcal{C}_1 = \{c_1\}$.
2. For $k = 2$ to $|M|$,
   - Let c_k be the furthest point from $\mathcal{C}_{k-1}$, i.e. maximizing $D(c_k, \mathcal{C}_{k-1})$.
   - Set $\mathcal{C}_k = \{c_k\} \cup \mathcal{C}_{k-1}$.
3. Output $\mathcal{C} = \langle c_1 \dots c_{|M|} \rangle$.

It is easy to see that each iteration of the greedy algorithm can be implemented in linear time $O(|M|)$, justifying the running time. Indeed, having selected points $\mathcal{C}_{k-1} = \{c_1 \dots c_{k-1}\}$, for each point $p \in M$ we only need to maintain the closest point $closest(p)$ in $\mathcal{C}_{k-1}$, i.e. the one achieving $D(p, closest(p)) = D(p, \mathcal{C}_{k-1})$. Assuming we have done this, c_k — the furthest point from $\{c_1 \dots c_{k-1}\}$ — is the point maximizing $D(p, closest(p))$, which takes linear time to find. To maintain $closest(p)$, initially we have $closest(p) = c_1$, and after selecting c_k we update $closest(p)$ to c_k only if $D(p, closest(p)) > D(p, c_k)$. These $|M|$ updates again take linear time per iteration.

Now, take any $2 \leq k \leq |M|$. The 2-competitiveness of $\mathcal{C}$ follows from the two claims below.

Claim 1: $d_{\mathcal{M}}(\mathcal{C}_k) = D(c_k, \mathcal{C}_{k-1})$, i.e. the closest pair of points in $\mathcal{C}_k$ includes c_k.
Proof: Assume $d_{\mathcal{M}}(\mathcal{C}_k) = D(c_i, c_j) < D(c_k, \mathcal{C}_{k-1})$, where $i < j < k$. Then $D(c_j, \mathcal{C}_{j-1}) = D(c_i, c_j) < D(c_k, \mathcal{C}_{k-1}) \leq D(c_k, \mathcal{C}_{j-1})$, i.e. c_k should have been added before c_j, a contradiction. □

Claim 2: $D(c_k, \mathcal{C}_{k-1}) \geq \frac{1}{2} \cdot \mathsf{opt}\text{-}d_{\mathcal{M}}(k)$.
Proof: Let $b_1 \dots b_k$ be the optimum code of cardinality k, i.e. $D(b_i, b_j) \geq \mathsf{opt}\text{-}d_{\mathcal{M}}(k)$ for some $i \neq j$.

Then the k open balls of radius $R = \frac{1}{2} \cdot \mathsf{opt}\text{-}d_{\mathcal{M}}(k)$ around the b_i's are all disjoint. Hence, at least one of these k balls does not contain any of the first $(k-1)$ selected points $c_1 \ldots c_{k-1}$. Say this is the ball around b_j. Hence, $D(b_j, \mathcal{C}_{k-1}) \geq R$. But c_k is the *furthest* point from $\mathcal{C}_{k-1}$, and, therefore, $D(c_k, \mathcal{C}_{k-1}) \geq D(b_j, \mathcal{C}_{k-1}) \geq R$. □

We next give an example of $\mathcal{M}$ with $\mathsf{opt}\text{-}r_{\mathcal{M}} = 2$. Let $M = \{w, x_1, x_2, y_1, y_2, z\}$, where $D(w, x_i) = 1$, $D(x_i, y_j) = 2$, $D(y_j, z) = 1$, $i, j = 1, 2$, and the other distances are the length of the shortest paths induced by the above assignments (see Figure 1). In particular, the two furthest points are w and z of distance 4, and the best 4-code is $\{x_1, x_2, y_1, y_2\}$ of minimum distance 2. In other words, $\mathsf{opt}\text{-}d_{\mathcal{M}}(2) = 4 = D(w, z)$ and $\mathsf{opt}\text{-}d_{\mathcal{M}}(4) = 2 = D(x_i, y_j)$, $i, j = 1, 2$. Now, for any incremental code $\mathcal{C} = \langle c_1, c_2, c_3, c_4 \rangle$, unless $\mathcal{C}_4 = \{x_1, x_2, y_1, y_2\}$, one of the pairwise distances in $\mathcal{C}_4$ will be 1, giving a gap of $2/1 = 2$. On the other hand, if $\mathcal{C}_4 = \{x_1, x_2, y_1, y_2\}$, then $D(c_1, c_2) = 2$, giving again a gap of $4/2 = 2$ for the 2-prefix of $\mathcal{C}$.

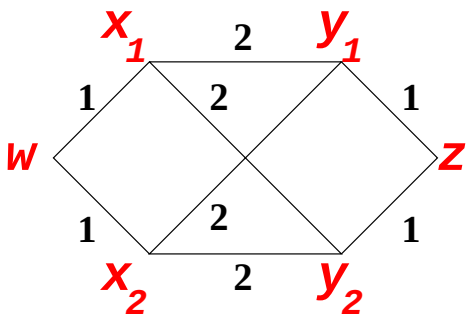

Figure 1: The "shortest path" closure of these distances define "bad" $\mathcal{M}$.

Finally, we show that it is NP-hard to construct an A-competitive code for $A < 2$ when given $\mathcal{M}$ as an input, even if $\mathsf{opt}\text{-}r_{\mathcal{M}} = 1$. We make a reduction from the MAXIMUM INDEPENDENT SET problem, which is known to be NP-complete [GJ79]. Given a graph $G = (V, E)$, we define a metric space $\mathcal{M} = (M, d)$, where $M = V$ and $D(i, j) = 1$ iff $(i, j) \in E$, and $D(i, j) = 2$ otherwise. Let $I = \{s_1 \ldots s_k\} \subseteq V$ be some maximum independent set of G. We claim that an optimal incremental code for $\mathcal{M}$ is 1-competitive, and should first list the elements of I (or any other maximum independent set) in any order, followed by the other elements in any order. Indeed, the code $\mathcal{C}$ constructed this way will have $d_{\mathcal{M}}(\mathcal{C}_i) = 2$ for $i \leq k$, and $d_{\mathcal{M}}(\mathcal{C}_i) = 1$ for $i > k$. On the other hand, the optimal code of cardinality i can have the minimum distance of 2 if and only if it is formed on the elements of some independent set in G, i.e. we must have $i \leq k$ (and this can be achieved). In other words, $\mathsf{opt}\text{-}d_{\mathcal{M}}(i) = 2$ for $i \leq k$ and $\mathsf{opt}\text{-}d_{\mathcal{M}}(i) = 1$ otherwise. To summarize, any 1-competitive code $\mathcal{C}$ for $\mathcal{M}$ induces the maximum independent set of G by looking at the largest largest i-prefix of $\mathcal{C}$ with $d_{\mathcal{M}}(\mathcal{C}_i) = 2$.

On the other hand, any code which is not 1-competitive for $\mathcal{M}$ must be 2-competitive. Hence, if we have a procedure that can produce A-competitive code for $\mathcal{M}$, where $A < 2$, this procedure must in fact produce an optimal 1-competitive code. But we just argued that in this case we can compute I — the largest independent set of G, which is NP-hard. □

Remark 1 *Notice that the greedy algorithm above is exactly the same as that of Gonzalez [G85] for the so called k-center problem. This is a just a coincidence, since our problems and the analysis are quite different.*

Remark 2 *Note, while the greedy algorithm is extremely efficient for generic metric spaces, we are mainly interested in the Hamming space F^n. For this space we cannot afford to go through the whole*

space, and would like our algorithms to be polynomial in $n \log(|F|) = \log(|M|)$. *We discuss an efficient* optimal *algorithm for this case when* $|F| \geq n$ *in Section 3, and efficient* competitive *algorithms over small fields in Section 4.*

Extending to compact spaces. Aside from the complexity considerations, we can extend much of the discussion above to infinite metric spaces. There are several things we need to ensure. First, the distances should be bounded, and for every finite k there should exist an optimal code of cardinality k. Additionally, the greedy algorithm that we gave in Theorem 1 makes perfect sense, as long as there exists a point which is furthest away from a given finite set of points. The discussion above suggests to use *compact* metric spaces, which satisfy each of the above requirements. For such infinite compact spaces, we replace exhaustive codes with countably infinite codes, and require every finite prefix of such a code to be "A-compatible" w.r.t. the best code of a given cardinality. We notice that Theorem 1 implies that there exists a countably infinite 2-competitive incremental code for any such metric space. In fact, the greedy algorithm is easily implementable for "nice" compact subsets in $\mathbb{R}^n$ (since on such sets we can compute the furthest point from a given set of points; of course, these computation could become less and less efficient once we introduce more points). For example, on the interval $[0, 1]$, and staring with $c_1 = 0$, our algorithm simply keeps subdividing the largest interval in half. Thus, after 2^k points, all the intervals will be of size $1/2^k$, but one interval will be of size $1/2^{k+1}$, giving a ratio $2^{k+1}/(2^k + 1) \to 2$. We will see in Section 3 that the best ratio for any competitive algorithm for $[0, 1]$ is in fact $\ln 4 = \log_e 4 \approx 1.386 < 2$.

3 Optimal Constructions

In this section we give two optimal constructions of incremental codes. The first construction is over the Hamming space F^n when $|F| \geq n$ (and F is a field), and shows that F^n is incrementally perfect. The second one is for the real interval $[0, 1]$ (which is not incrementally perfect).

3.1 Optimal 1-Competitive Code for Hamming space F^n, $|F| \geq n$

We give a very simple costruction of an optimal incremental code based on *Reed-Solomon Codes* (RS-codes), which we briefly recall now. Let F be a field of size $|F| = q$. RS-code of dimension k and block length n over F (where $|F| = q \geq n$) maps elements of F^k into codewords over F^n via the following procedure. Let $\alpha_1 \dots \alpha_n$ be arbitrary distinct elements of F. Given $a = (a_0 \dots a_{k-1}) \in F^k$, assign a polynomial $p_a(x) = \sum_{i=0}^{k-1} a_i x^i$ of degree at most $(k-1)$, and output the codeword $(p(\alpha_1) \dots p(\alpha_n)) \in F^n$. Since any two distinct polynomials of degree at most $(k-1)$ can agree on at most $(k-1)$ points in F, the distance of the RS-code at least (in fact, exactly) $d = n - k + 1$. On the other hand, the classical singleton bound says that *any* code of dimension k (i.e., q^k codewords) must have minimal distance at most $(n - k + 1)$, achieved by the corresponding RS-code. Hence, RS-codes are *optimal* codes of dimension k, and, in particular, $\mathsf{opt}\text{-}d_{\mathcal{M}}(q^k) = n - k + 1$ (where $\mathcal{M} = F^n$) when the size of the field $q \geq n$.

Letting $F^{k-1}[x]$ to denote the set of polynomials of degree at most $(k-1)$, we can view the RS-code of dimension k as mapping an element of $F^{k-1}[x]$ into F^n. Viewed this way, we observe that $F^0[x] \subseteq F^1[x] \dots \subseteq F^{n-1}[x]$, which allows us to to view the RS-code of dimension $(k-1)$ and optimal distance $(n-k+2)$ as a subset of the RS-code of rate k and optimal minimal distance $(n-k+1)$. Thus, if we first encode (i.e., evaluate at n points) the polynomials of degree 0, followed by the polynomials of degree 1 and so on, we see that the minimal distance of our *incremental* code slowly decreases from

n to $(n-1)$, ..., all the way to 1. More specifically, at the time we are encoding polynomials of degree k, our current code has minimal distance $(n-k+1)$ (being part of the RS-code of dimension k), which is optimal by the singleton bound since our current code has more than q^{k-1} elements (as we already listed q^{k-1} polynomials of degree at most $(k-1)$).

Thus, we showed that the Hamming space F^n has a 1-competitive exhaustive code $\mathcal{C} = \langle c_0, \ldots, c_{N-1} \rangle$, where $N = q^n$. To efficiently compute c_i, we notice the following. If we write the elements of F as numbers $0, \ldots, (q-1)$ (0 being the "zero" of F), and then interpret the representation of an integer $i \in \{0, \ldots, N-1\}$ base $|F|$ as a string $a(i) \in F^n$, then listing i in the increasing order corresponds to the lexicographic order of the $a(i)$'s, which also lists the polynomials $p_{a(i)}$ in the order of increasing degrees, as needed. To summarize, we get

Theorem 2 *F^n is incrementally perfect when $|F| = q \geq n$ and F is a field. In particular, the incremental code $\mathcal{C} = \langle c_0, \ldots, c_{N-1} \rangle$, where $c_i = (p_{a(i)}(\alpha_1), \ldots, p_{a(i)}(\alpha_n)) \in F^n$, is exhaustive and 1-competitive. Moreover, for $1 \leq k \leq n$ and when $q^{k-1} \leq i < q^k$, the minimal distance of $\{c_0, \ldots, c_i\}$ is $(n-k+1)$.*

Practical Discussion: We notice that the procedure of computing c_i is extremely simple, practical and efficient (at most quadratic in n and $\log |F|$), and does not need to keep any state as i grows (unlike, for example, the greedy algorithm for generic metric spaces). For several obvious examples, see Appendix A.

3.2 Optimal Code for $[0,1]$

An incremental code over $[0,1]$ is simply a sequence of points $\mathcal{C} = \langle p_1, p_2, \ldots \rangle$. If we let $q_1^i \ldots q_i^i$ denote $p_1 \ldots p_i$ in the increasing order (so that $q_1^i \leq q_2^i \ldots \leq q_i^i$), then after i steps $[0,1]$ is split into $(i+1)$ intervals $I_0 = [0, q_1^i]$, $I_2 = [q_1^i, q_2^i]$, ..., $I_i = [q_i^i, 1]$. Clearly, the minimal distance of $\mathcal{C}_i$ is $d(\mathcal{C}_i) = \min(|I_1|, \ldots, |I_{i-1}|))$, while the optimum distance is $\mathsf{opt}\text{-}d(i) = 1/(i-1)$ (by spreading the points uniformly). When adding p_{i+1} we simply subdivide one of the I_j's into two subintervals. If we assume that $p_1 = 0$ and $p_2 = 1$ (which will happen in our solution and will be the "worst case" in the lower bound proof), then the "border" intervals I_0 and I_{i+1} disappear, and our objective is to place the points $p_3, p_4, \ldots$ on $[0,1]$ in such a manner that the length of the *smallest* interval after each p_i is as close to $1/(i-1)$ as possible. We notice that the dual "maximal interval" version of the latter problem — make the *largest* interval as close to $1/(i-1)$ as possible — is a well known *dispersion problem* (see [DT97, C00, M99]). While our lower bound and its proof will be somewhat different for our "minimal interval" version, it will turn out that the optimal sequence for both versions will be the *same*, which is not at all clear a-priori.

Let $H(k) = (1 + \frac{1}{2} + \ldots + \frac{1}{k})$ denote the k^{th} harmonic series.

Lemma 1 *Unless $A \geq 2 \cdot [H(2i) - H(i+1)]$, no incremental code of $(2i+1)$ points in $[0,1]$ is A-competitive.*

Proof: Consider a code of $2i+1$ points in $[0,1]$ with competitive ratio A. For every $j \leq i$, consider the distances between adjacent points after placing the first j points. Let $\ell_1^j \leq \ell_2^j \leq \ldots \ell_{j-1}^j$ be these distances, sorted in increasing order. We need the following claim:

Claim: $\ell_k^j \leq \ell_{k+2}^{j+1}$ for $1 \leq k \leq j-2$.

Proof: Adding a point (in this case, $(j+1)$-st point) can either add one more distance to the list of interval distances (if the new point is the rightmost or the leftmost), or it can remove one length from

the list, replacing it with two others (if the new point lies between two old points). In either case, there are at most two new lengths that are added to list. This means that among the first $k+2$ lengths on the new list, there are at least k lengths that were already on the old list before we added the last point. Hence, the $k+2$'nd smallest length on the new list cannot be smaller than the k'th smaller length on the old list. □

By iterating the above claim, we get for all $0 \le j \le i-1$, $\ell_1^{2i+1-j} \le \ell_{1+2j}^{2i+1}$. Notice, ℓ_1^{2i+1-j} is the length of the smallest interval after adding $(2i+1-j)$ points. Since our code is A-competitive (and since the optimal arrangement of $2i+1-j$ points has distance $1/(2i-j)$), we must have $\frac{1/(2i-j)}{\ell_1^{2i+1-j}} \le A$, which means that $\ell_{1+2j}^{2i+1} \ge \ell_1^{2i+1-j} \ge 1/(A(2i-j))$. Summing the last inequality for $j = 0 \ldots i-1$, we get

$$\sum_{j=0}^{i-1} \ell_{1+2j}^{2i+1} \ge \frac{1}{A} \cdot \left(\frac{1}{2i} + \frac{1}{2i-1} + \ldots + \frac{1}{i+2} \right) = \frac{1}{A} \cdot [H(2i) - H(i+1)] \tag{1}$$

On the other hand, since $\ell_{1+2j}^{2i+1} \le \ell_{2+2j}^{2i+1}$ and all the $2i$ intervals sum to at most 1, so we get

$$\sum_{j=0}^{i-1} \ell_{1+2j}^{2i+1} \le \sum_{j=0}^{i-1} \left(\frac{\ell_{1+2j}^{2i+1} + \ell_{2+2j}^{2i+1}}{2} \right) \le \frac{1}{2} \tag{2}$$

Combining Equation (1) and Equation (2), we get $A \ge 2 \cdot [H(2i) - H(i+1)]$. □

Since $2 \cdot [H(2i) - H(i+1)] \approx 2 \ln \left(\frac{2i}{i+1} \right) \stackrel{i \to \infty}{\longrightarrow} \ln 4$, we get

Corollary 2 *If $\mathcal{C}$ is an infinite A-competitive code, then $A \ge \ln 4 \approx 1.386$.*

We now show an incremental code achieving the bound above. We let $p_0 = 0$, $p_1 = 1$, and explicitly tell the lengths of the i intervals after the first $(i+1)$ points. They are (in increasing order): $\log_2(1 + \frac{1}{2i-1}), \log_2(1 + \frac{1}{2i-2}), \ldots, \log_2(1 + \frac{1}{i})$. Notice, $\sum_{j=1}^{i} \log_2(1 + \frac{1}{2i-j}) = \log_2(\prod_{j=1}^{i} \frac{2i-j+1}{2i-j}) = \log_2(\frac{2i}{i}) = 1$, indeed. Also, for $i = 1$ our only interval is indeed of size $1 = \log_2(1 + \frac{1}{1})$. To add the $(i+2)^{nd}$ point, we subdivide the currently largest interval of size $\log_2(1 + \frac{1}{i})$ into two intervals of sizes $\log_2(1 + \frac{1}{2i})$ and $\log_2(1 + \frac{1}{2i+1})$ (again, arithmetic works), as claimed. We see that the length of the smallest interval after $(i+1)$ points is $\log_2(1 + \frac{1}{2i-1}) \ge \frac{1}{i \ln 4}$ (the latter is easy to check), proving that this sequence have competitive ratio $\ln 4$.

Remark 3 *The above argument and construction can be adjusted to the case of the closed circle S^1 (where distance is the shortest arc of the circle), and give the same competitive ratio. The first two points of the code are put diametrically opposite to each other, and then we interleave the "interval" construction above on the lower and upper semi-circles. The lower bound extends as well.*

4 Hamming Space (Error-Correcting Codes)

In this section we discuss the Hamming space F^n. We refer the reader to [MS81] for more information on some of the facts we use in this chapter. Recall, we showed in Theorem 2 that F^n is incrementally perfect when $|F| = q \ge n$ (and F is a field). We now consider the more difficult case when $|F| = q \ll n$ (in particular, binary). As we will see, obtaining tight bounds for this case is expected to be harder (see below), but first let us recall some terminology that we will need later.

A code with K codewords and minimal distance d over F is said to have *rate* $\tau = \log_q K/n$, *dimension* $k = \log_q K$ and *relative distance* $\delta = d/n$. We omit the subscript to the space from the quantities opt-r and opt-d when the Hamming space is clear from the context, and otherwise write $\mathsf{opt}\text{-}r(K;\ q, n)$ and $\mathsf{opt}\text{-}d(K;\ q, n)$ to emphasize the space. We let $\mathsf{opt}\text{-}\delta(\tau;\ q, n) = \frac{1}{n} \cdot \mathsf{opt}\text{-}d(q^{\tau n};\ q, n)$ be the largest possible relative distance of a code of rate τ over F. We let $V_q(R;\ n)$ denote the volume of an n-dimensional sphere of radius R in F^n, and notice that asymptotically, we have $\frac{1}{n} \cdot \log_q V_q(\alpha n;\ n) \approx H_q(\alpha) = \alpha \log_q(q-1) - \alpha \log_q \alpha - (1-\alpha)\log_q(1-\alpha)$, where $H_q(\cdot)$ above is the q-ary entropy function (in particular, $V_q(\alpha n;\ n) \leq q^{nH_q(\alpha)}$).

As a first observation, we notice below that the Hamming space is not incrementally perfect (we leave the proof for Appendix B):

Lemma 3 *If $q < n$, then the Hamming space $[q]^n$ is not incrementally perfect, unless $q = 2, n = 4$.*

How good is the optimal code? One small problem with the notion of competitive ratio over small alphabets, is that we need to compare the performance of the code with that of the optimal code of the same rate. For codes over small alphabets, we only have bounds on the minimal distance of the optimal code, rather than a closed-form formula. Hence, the competitive ratio that we can prove depends not only on the performance of the code in question, but also on the quality of these bounds.

On a brighter side, the discrepancy between the known upper- and lower-bounds on the optimal distance as a function of rate is at most a small constant factor. In fact, the ratio between the Hamming bound (an upper-bound) and the Gilbert-Varshamov bound (a lower-bound) is a factor of 2 "in spirit". To see that, recall that the Hamming bound says that for any code with K codewords and minimum distance d over $[q]^n$, it holds that $K \cdot V_q(d/2;\ n) \leq q^n$, i.e. $\mathsf{opt}\text{-}d(K;\ q, n) \leq 2V_q^{-1}(q^n/K;\ n)$. The Gilbert-Varshamov bound, on the other hand, says that there exists a K-word code with minimum distance d satisfying $K \cdot V_q(d;\ n) \geq q^n$, i.e. $\mathsf{opt}\text{-}d(K;\ q, n) \geq V_q^{-1}(q^n/K;\ n)$.

In principle, this means that when we use the Hamming bound as our estimate for the performance of the optimal code, we only lose a factor of two (or less). However, notice that when we use that bound, we usually use some estimate for V_q^{-1} (since working with V_q^{-1} itself is too hard), so we may lose some small additional factor there (see Section 4.3 for an example).

What is an "efficient" construction? As opposed to the generic case, where we are given the entire metric space as input and need to produce as output the "code" itself (as a list of points), in the case of the Hamming space we usually think of entire space as being exponential in the relevant parameters, and we think of the code as having some implicit small representation. What we may require in terms of efficiency is to have a representation of the code whose length is polynomial in n and $\log q$, and an efficient procedure that given this representation and an index i, produces c_i, the i'th codeword. For example, viewed in this light, a random code is not an "efficient construction", but a random linear code is.

Constructions. We now turn to the question of efficient constructions of incremental codes. As a most trivial construction, consider a regular code with K codewords, minimum distance d and relative distance $\delta = d/n$. How well does it perform as an incremental code (under arbitrary ordering)? Without any additional knowledge about the code, the best competitive ratio we can get is $n/d = 1/\delta$. Still, if we take a family of *asymptotically good* codes[1], we get a family of incremental codes with constant rate and constant competitive ratio. Of course, this simplistic construction has several shortcomings. First, there is a pretty stringent tradeoff between the rate and the relative distance of the code, so we will

[1]Recall, family of codes $\{\mathcal{C}^n\}_{n \in \mathcal{N}}$ is asymptotically good if both the relative distance and the rate of $\mathcal{C}^n$ is $\Omega(1)$.

either sacrifice the rate (make the code very sparse), or the competitive ratio. In particular, if the rate is close to 1, the completive ratio tends to ∞. Secondly, even on small prefixes our code can have the same minimal distance d, i.e. the distance does not necessarily "gradually decrease". Because of that, there is no point in using more sophisticated bounds than n on the optimal code's minimal distance. Thus, this approach does not address the essence of the problem at all.

Therefore, we use more sophisticated techniques that will give us better tradeoffs between the rate and the competitive ratio of incremental codes. Specifically, we examine three efficient constructions: (1) using algebraic-geometric codes (AG-codes) as a natural generalization of the RS-codes to small fields, (2) using concatenation theorem to reduce the alphabet size, and (3) using random linear codes. The latter construction will let us achieve our ultimate goal: have an absolute constant competitive ratio (slightly more than 2), even when the rate is $1 - o(1)$.

4.1 Algebraic-Geometric Codes

Algebraic-Geometric Codes (AG-codes) are natural extensions of the RS-codes to small fields. Detailed treatment of AG-codes is beyond the scope of this paper, so we informally concentrate on the essentials only (see [S93] for more information). Rather than talking about polynomials of degree at most α which can be evaluated at n points in the field, AG-codes deal with algebraic functions with at most α "poles" at the "point of infinity" which can be evaluated at n "rational points" of the function field. In both cases, the valuation map returns n elements of F, and a given polynomial/algebraic function can have at most α zeros. We let $L(\alpha, \infty)$ denote the space of such functions, which turns out to be a linear space over F. The famous Riemann-Roch theorem says that the dimension of this space is at least $\alpha - g + 1$, where g is the "genus" of the algebraic field (for the RS-codes we can achieve $g = 0$, but for smaller fields g cannot be very small; see below). All together, AG-codes given by the space $L(\alpha, \infty)$ have the following parameters: the dimension $k \geq \alpha - g + 1$, the distance $d \geq n - \alpha$. Like with the RS-codes, we observe that $L(0; \infty) \subseteq L(1; \infty) \subseteq \ldots \subseteq L(n-1; \infty)$, which implies that AG-codes of increasing pole orders at infinity define an incremental code, the first q^k codewords of which have minimal distance at least $(n - k - g + 1)$. Since the singleton bound still says that the optimum distance is at most $(n - k + 1)$, we get

Theorem 3 *The competitive ratio of AG-codes of dimension k (listed in the order specified above) is at most $\frac{n-k+1}{n-k-g+1}$. In particular, setting $k = n - 2g + 1$ gives an incremental code with q^k codewords and competitive ratio at most 2.*

While the main advantage of the AG-codes is the fact that they are defined on small (e.g. constant size) alphabets, we briefly point out their limitations. In particular, the bound above is meaningful only when $k < n - g$, i.e. the rate can be at most $(1 - \frac{g}{n})$. It is known that $g \geq n/(\sqrt{q} - 1)$,[2] so the maximal rate we can hope to achieve is roughly $(1 - \frac{1}{\sqrt{q}-1})$.

4.2 Concatenation Theorem

We next address a general method of constructing an incremental code over small alphabet from the one over a large alphabet and a good regular error-correcting code over a small alphabet. This method is completely analogous to the one used when constructing regular codes over small alphabets, and is called the *concatenation* of codes.

[2] When n grows w.r.t. q, and this bound is tight for certain q's. The general bound is $g \geq (n - q - 1)/(2\sqrt{q})$.

Let $\mathcal{C} = \langle c_1 \dots c_K \rangle$ be an incremental code in $[q]^n$, with competitive ratio A and rate $\tau = (\log_q K)/n$. Let $T = \{t(1) \dots t(q)\}$ be a regular code in $[q_2]^{n_2}$ ($q_2 \ll q$), with distance d_2, relative distance $\delta_2 = d_2/n_2$, and rate $\tau_2 = (\log_{q_2} q)/n_2$. An incremental code $\mathcal{C}^* = \mathcal{C} * T = \langle c^*(1) \dots c^*(K) \rangle \subseteq [q_2]^{n \cdot n_2}$, the *concatenation* of $\mathcal{C}$ and T, is defined as follows. If we write the i-th codeword of $\mathcal{C}$ as $c_i = c_{i,1} \dots c_{i,n} \in [q]^n$, and interpret each $c_{i,j}$ as an integer in $\{1 \dots q\}$, then the i-th codeword of $\mathcal{C}^*$ is $c^*_i = t(c_{i,1}) \dots t(c_{i,n}) \in [q_2]^{nn_2}$. The code $\mathcal{C}$ is called the "outer code", and T is called the "inner code".

Theorem 4 *$\mathcal{C}^*$ is an incremental code in $[q_2]^{nn_2}$ with K codewords, rate $\tau^* = (\log_{q_2} K)/(nn_2) = (\log_q K \cdot \log_{q_2} q)/(nn_2) = \tau\tau_2$, and competitive ratio A^* satisfying:*

$$A^* \leq \frac{A}{d_2} \cdot \max_{i \leq K} \frac{\mathsf{opt}\text{-}d(i;\ q_2, nn_2)}{\mathsf{opt}\text{-}d(i;\ q, n)} = \frac{A}{\delta_2} \cdot \max_{\rho \leq \tau} \frac{\mathsf{opt}\text{-}\delta(\rho\tau_2;\ q_2, nn_2)}{\mathsf{opt}\text{-}\delta(\rho;\ q, n)} \tag{3}$$

Proof: Take the i-prefix $\mathcal{C}^*_i$ of $\mathcal{C}^*$. We claim that $d(\mathcal{C}^*_i) \geq d(\mathcal{C}_i) \cdot d_2$, which is clear from the construction of $\mathcal{C}^*$. Using also A-competiveness of $\mathcal{C}$ (i.e. $\mathsf{opt}\text{-}d(\mathcal{C}_i) \leq A \cdot d(\mathcal{C}_i)$), we get

$$A^* = \max_{i \leq K} \frac{\mathsf{opt}\text{-}d(i;\ q_2, nn_2)}{d(\mathcal{C}^*_i)} \leq \max_{i \leq K} \frac{\mathsf{opt}\text{-}d(i;\ q_2, nn_2)}{\mathsf{opt}\text{-}d(i;\ q, n)} \cdot \frac{\mathsf{opt}\text{-}d(i;\ q, n)}{d(\mathcal{C}_i) \cdot d_2} \leq \frac{A}{d_2} \cdot \max_{i \leq K} \frac{\mathsf{opt}\text{-}d(i;\ q_2, nn_2)}{\mathsf{opt}\text{-}d(i;\ q, n)}$$

□

Clearly, we can try to substitute some known upper (resp. lower) bounds in place of $\mathsf{opt}\text{-}\delta(i;\ q_2, nn_2)$ (resp. $\mathsf{opt}\text{-}\delta(i;\ q, n)$), to get a more algebraic expression in the bound above. For example, in the asympototic sense we can use the Hamming bound in the numerator, and the Gilbert-Varshamov bound in the denominator, and get

$$\max_{\rho \leq \tau} \frac{\mathsf{opt}\text{-}\delta(\rho\tau_2;\ q_2, nn_2)}{\mathsf{opt}\text{-}\delta(\rho;\ q, n)} \gtrsim \max_{\rho \leq \tau} \frac{2 \cdot H^{-1}_{q_2}(1 - \rho\tau_2)}{H^{-1}_q(1 - \rho)} = \frac{2 \cdot H^{-1}_{q_2}(1 - \tau\tau_2)}{H^{-1}_q(1 - \tau)}$$

(recall, H_q is the q-ary entropy function). However, such generic bound are often not much easier to work with, and different such bounds could be more convenient for different constructions.

Below we illustrate the bound from Theorem 4 for the case where the outer code is the incremental RS-code constructed in Section 3.1, which is perhaps the most attractive case to consider. Recall, from Section 3.1 that these incremental codes are 1-competitive, can be used with any rate $\tau \leq 1$, and have the restriction that $q \geq n$.

Corollary 4 *Let $\mathcal{C}$ be the 1-competitive RS-code of rate τ, and T be as before. Then $\mathcal{C}^*$ has rate $\tau^* = \tau\tau_2$ and competitive ratio*

$$A^* \leq \frac{1 - \tau\tau_2}{\delta_2(1 - \tau)} \tag{4}$$

Proof: We notice that for the RS-code we have $A = 1$ and $\mathsf{opt}\text{-}\delta(\rho;\ q, n) = 1 - \rho + 1/n$. Now the most convenient bound to use for $\mathsf{opt}\text{-}\delta(\rho\tau_2;\ q_2, nn_2)$ seems to be the (very loose in general) singleton bound, which says that $\mathsf{opt}\text{-}\delta(\rho\tau_2;\ q_2, nn_2) \leq 1 - \rho\tau_2 + 1/(nn_2)$. Using Equation (3) now, we get

$$\max_{\rho \leq \tau} \frac{1 - \rho\tau_2 + \frac{1}{nn_2}}{1 - \rho + \frac{1}{n}} \ \leq \ \max_{\rho \leq \tau} \frac{1 - \rho\tau_2}{1 - \rho} \ = \ \frac{1 - \tau\tau_2}{1 - \tau}$$

□

We notice the tradeoff that we obtain. In particular, $\frac{1-\tau\tau_2}{\delta_2} \geq \frac{1-\tau_2}{\delta_2} \geq 1$ (the latter part follows from the singleton bound applied to T). Thus, our guarantee on A^* cannot be better than $1/(1-\tau)$. This implies that if we want a constant competitive ratio from $\mathcal{C}^*$, we cannot make $\tau = 1 - o(1)$. In other words, even though we can extend the RS-code to all the rates up to 1, our analysis can no longer provide a constant guarantee on A^*. On a positive note, we can make the rate $\tau^* = \tau\tau_2$ of $\mathcal{C}^*$ arbitrarily close to 1 (at the expense of A^*). Finally, it is also interesting to compare the bound in Equation (4) with the trivial bound we get by simply viewing $\mathcal{C}^*$ as a code of minimal relative distance $\delta^* = \delta_2(1-\tau)$. We see that we would get the ratio $1/(\delta_2(1-\tau))$, which is a factor $(1-\tau\tau_2)$ worse than our bound.[3]

Justeson Codes: Justeson codes, which were the first known asymptotically good *binary codes*, are also constructed using the concatenation theorem: the outer code is again any RS-code, while the inner codes are special "good average distance" codes of rate 1/2. It is well known (see [MS81]) that Justeson codes can achieve any rate $\tau < 1/2$, while having relative distance at least $(1-2\tau)H_2^{-1}(1/2) \approx 0.1(1-2\tau)$. A straightforward extension of our concatenation theorem shows that such a code could be ordered into an incremental code of rate τ and competitive ratio at most $10(1-\tau)/(1-2\tau)$. Again, we save a factor $(1-\tau)$ over the trivial bound (order arbitrarily).

4.3 Random Linear Codes

We saw that the explicit (and efficient) constructions from the previous sections failed to achieve a constant competitive ratio for rates $(1-o(1))$. On the other hand, Theorem 1 shows the existence (and inefficient construction) of an exhaustive 2-competitive codes for any $[q]^n$. In this section we show that, with high probability, a *random linear code* will achieve a competitive rate $2(1+\epsilon)$ (for arbitrarily small ϵ, and possibly better if of understanding of optimal codes will improve), even for rate $(1-o(1))$ (specifically, dimension up to $n - \Theta(\log_q n)$). This gives an efficient (albeit randomized) procedure to generate competitive and almost exhaustive incremental codes.

Recall that a (standard) random linear code of dimension k in $[q]^n$, is the set of words which are spanned by the rows of a $k \times n$ random matrix G over $[q]$. As we are interested in *ordered* codes, we consider these words in a canonical order of their coefficients. Namely, for a given matrix G, the order between two codewords $c = xG$ and $c' = x'G$ is determined by the lexicographic order between x and x'.[4] We note that with this ordering, all the codewords that are spanned by the first i rows of G, appear before all the codewords that depend also the $i+1$'st row. This means that for any $m \leq k$, the prefix $\mathcal{C}_{q^m}$ is itself a (random) linear code.

We first recall an easy lemma that bounds the minimum distance of a random linear code.

Lemma 5 *For $k \leq n$, let G be a random $k \times n$ q-ary matrix, and let $\mathcal{C} = \mathcal{C}(G) = \{c_1 \ldots c_{q^k}\}$ be the ordered linear code that is spanned by G. Then for every $\epsilon < 1 - \frac{1}{q}$, we have*

$$\Pr\left[d(\mathcal{C}) < \epsilon n\right] < 2^{k - n(1-H_q(\epsilon))}$$

Proof: Let $S(\epsilon n)$ be the sphere of radius ϵn around the origin in Hamming space $[q]^n$. We know that $S(\epsilon n)$ contains $V_q(\epsilon n;\ n) \leq q^{nH_q(\epsilon)}$ words. For a fixed k-vector $x \neq \bar{0}$, we have $\Pr_G[xG \in S(\epsilon n)] \leq q^{nH(\epsilon)-n}$. Using the union bound, we get

$$\Pr\left[d(\mathcal{C}) < \epsilon n\right] \ = \ \Pr[\exists x \neq \bar{0},\ xG \in S(\epsilon n)] \ < \ q^k \cdot q^{-n(1-H(\epsilon))}$$

[3] In fact, a tighter gap implied by Theorem 4 is $\mathsf{opt}\text{-}\delta(\tau\tau_2;\ q_2, nn_2)$, which is usually much less than $1-\tau\tau_2$.

[4] Notice, this is the same lexicographic order we used with the RS-codes in Section 3.1.

□

Corollary 6 *Let $k < n$, $\delta > 0$, and G be a random $k \times n$ q-ary matrix. With probability of at least $1 - \delta$ (over the choice of G), the minimum distance of $\mathcal{C}(G)$ is at least $nH_q^{-1}\left(\frac{n-k-\log_q(1/\delta)}{n}\right)$.*

Below we prove that a random linear code has competitive ratio of at most $2(1+\epsilon)$ for rates up to $1 - \Theta(\frac{\log_q n}{n})$. For this proof, we first recall a somewhat weak variant of the Hamming bound, and one fact about the (inverse of the) q-ary entropy function H_q.

Hamming bound. For any q and $k \leq n$, $\mathsf{opt}\text{-}d(q^k;\ q, n) \leq 2nH_q^{-1}\left(\frac{n-k-1+\log_q n}{n}\right)$.

Proposition 7 *For any constant $\epsilon > 0$, there exists a constant "threshold" $\rho = \rho(\epsilon) > 0$, so that for all $\delta \leq \rho$, all $z \geq 2\delta/\epsilon$ and all $q \geq 2$, it holds that $H_q^{-1}(z+\delta)/H_q^{-1}(z) < 1 + \epsilon$.*

Theorem 5 *For any q, any constant $\epsilon > 0$, any large enough n, and any $k \leq n - (2 + \frac{6}{\epsilon})\log_q n$, a random linear code of dimension k in $[q]^n$ has competitive ratio $A \leq 2(1+\epsilon)$, with probability at least $1 - \frac{1}{n}$.*

Proof: Let G be a random $k \times n$ q-ary matrix and let $\mathcal{C} = \mathcal{C}(G)$ be the ordered linear code spanned by G. For each integer $m \leq k$, Corollary 6 with $\delta = 1/n^2$ tells us that with probability at least $1 - 1/n^2$, the minimum distance of $\mathcal{C}_{q^m}$ (i.e., the code spanned by the first m rows of G) is at least

$$d_m = nH_q^{-1}\left(\frac{n - m - 2\log_q n}{n}\right)$$

Taking the union bound, we conclude that with probability at least $1 - k/n^2 > 1 - 1/n$, the above holds for all $1 \leq m \leq k$. This, in turn, implies that for any m and any $i \in \left\{q^{m-1}+1, \ldots, q^m\right\}$, the minimum distance of $\mathcal{C}_i$ is at least d_m.

On the other hand, the Hamming bound tells us that the minimum distance of the optimal q^{m-1}-word code cannot be more than

$$d^*_{m-1} = 2nH_q^{-1}\left(\frac{n - (m-1) - 1 + \log_q n}{n}\right) \ = \ 2nH_q^{-1}\left(\frac{n - m + \log_q n}{n}\right)$$

This implies that also for every $i \in \left\{q^{m-1}+1, \ldots, q^m\right\}$, the minimum distance of the optimal i-word code cannot be more than d^*_{m-1}. We conclude that with probability at least $1 - 1/n$, the competitive ratio of $\mathcal{C}$ is bounded below by $\max_m(d^*_{m-1}/d_m)$.

Fix any $m \leq k$, and denote $\delta = \frac{3\log_q n}{n}$ and $z = \frac{n-m-2\log_q n}{n}$. Since $m \leq k \leq n - (2 + \frac{6}{\epsilon})\log_q n$, it follows that $z \geq (\frac{6}{\epsilon}\log_q n)/n = 2\delta/\epsilon$. Also, for large enough n we have $\delta \leq \rho(\epsilon)$ (where ρ is the "threshold" function from Proposition 7), we can use Proposition 7 to conclude that

$$\frac{d^*_{m-1}}{d_m} = \frac{2nH_q^{-1}(z+\delta)}{nH_q^{-1}(z)} \leq 2(1+\epsilon)$$

As the inequality above holds for any $m \leq k$, this completes the proof of the theorem. □

References

[C00] B. Chazelle. The Discrepancy Method: Randomness and Complexity. *Cambridge University Press*, 2000.

[DT97] M. Drmota, R. Tichy. Sequences, Discrepancies, and Applications *Lecture Notes in Mathematics 1651*, Springer, Berlin, 1997.

[GJ79] M. Garay, D. Johnson. Computers and Intractability. *W.H. Freeman and Company*, New York, 1979.

[G85] T. Gonzalez. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science*, (38)293–306, 1985.

[MS81] F. MacWilliams, J. Sloane. Theory of Error-Correcting Codes, Amsterdam, 1981.

[M99] J. Matousek. Geometric Discrepancy (An Illustrated Guide). *Springer-Verlag*, Berlin, 1999.

[S93] H. Stichtenoth. Algebraic Function Fields and Codes. *Springer-Verlag*, Berlin, 1993.

A Simple Examples for the Motivating Problem

Consider the problem of assigning account numbers to Internet shoppers. For for any n, we can select any prime power (since F is a field) $q \geq n$ to be $|F|$, and be able to serve the maximal number of customers, q^n, while having the property than when the number of customers is less than q^k, all account numbers have at least $(n-k+1)$ distinct symbols. Here are some obvious settings.

For example, if $q = n = 9$, we get practical scheme for $9^9 = 387,420,489$ customers with 9-digit accounts over digits $1 \ldots 9$ (say, 0 is a special character when entering on the phone), the first $4,782,969$ (resp. $43,046,721$) of which have at least 3 (resp. 2) distinct digits. And if 11 digits are acceptable, we can have 11^{11} (which is more than we will ever need) account numbers (say, over $0 \ldots 9, *$), the first 20 (resp. 200) million of which have at least 5 (resp. 4) distinct digits. Finally, if we can use 25 English letters as characters, even 7-character account numbers let us handle the world population (more than 6 billion numbers), the first 10 (resp. 250) million of which have at least 3 (resp. 2) distinct characters.

B Proof of Lemma 3

When $q < n$, we do not know the exact behavior of optimal codes. Luckily, we can establish the claim by looking only at the first $q+1$ points of the code, when we still know what the optimum looks like. So let $q < n$ and assume that $[q]^n$ is incrementally perfect. Since the words $1^n \ldots q^n$ have pairwise distance n, we have that $\mathsf{opt}\text{-}d(q) = n$. Hence, if $[q]^n$ has a 1-competitive incremental code $\mathcal{C}$, it must be that $d(\mathcal{C}_q) = n$. Namely, any two of the first q words of $\mathcal{C}$ must differ in all the coordinates $1 \leq i \leq n$. In fact, we can assume w.l.o.g. that the words $0^n \ldots (q-1)^n$ are the first q words in $\mathcal{C}$.[5] This means, however, that the $(q+1)$'st word of $\mathcal{C}$ must agree with one of the first q words in at least $\left\lceil \frac{n}{q} \right\rceil$ positions, implying that $d(\mathcal{C}_{q+1}) \leq n - \left\lceil \frac{n}{q} \right\rceil$.

[5] This is true since we can always permute the symbols in coordinate i of all the words of $\mathcal{C}$, without changing any of the distances of the code.

On the other hand, we now show that the optimal $(q+1)$-word code in $[q]^n$ has minimum distance at least $n - \left\lceil \frac{2n}{q(q+1)} \right\rceil$. Consider a $(q+1) \times n$ matrix, whose rows would correspond to $q+1$ codewords. We describe how to fill this matrix so that every two rows would agree in at most $\lceil 2n/(q(q+1)) \rceil$ coordinates. Specifically, we fill the columns of this matrix in "chunks" of $\binom{q+1}{2}$ at a time, making sure that in each "chunk" every two rows agree in at most one coordinate. This is done as follows: the $\binom{q+1}{2}$ columns in each "chunk" correspond to all pairs of distinct indices $1 \leq i < j \leq q+1$. Namely, the column $v(i,j)$ corresponding to the pair (i,j) has symbol q in positions i and j, and all the other symbols $1 \ldots (q-1)$ in the other $(q-1)$ positions of $v(i,j)$ (in arbitrary order). So within column $v(i,j)$, the only two positions that agree are positions i and j. It follows that within the current "chunk", any two rows i and j agree only in the column $v(i,j)$. And as there at most $\lceil n/\binom{q+1}{2} \rceil = \left\lceil \frac{2n}{q(q+1)} \right\rceil$ such "chunks", any two rows agree in at most $\left\lceil \frac{2n}{q(q+1)} \right\rceil$ coordinates.

Since we assume that $\mathcal{C}$ is 1-competitive, we must have

$$n - \left\lceil \frac{2n}{q(q+1)} \right\rceil \;\leq\; \mathsf{opt}\text{-}d(q+1) \;=\; d(\mathcal{C}_{q+1}) \;\leq\; n - \left\lceil \frac{n}{q} \right\rceil$$

i.e. $\left\lceil \frac{n}{q} \right\rceil \leq \left\lceil \frac{2n}{q(q+1)} \right\rceil$. It is not hard to see that the only pair $n > q$ that satisfies this inequality is $n = 4$ and $q = 2$, i.e. no other space can be incrementally perfect. As for $\{0,1\}^4$, it is indeed incrementally perfect via the optimal code $\{0000, 1111,$ all words with two 1's, all the rest$\}$.

Remark 4 *Notice that a tighter analysis shows that for any q, if n is large enough, any code of size $(q+1)$ in $[q]^n$ has competitive ratio at least $1 + 1/\binom{q+2}{2}$. For example, for $q = 2$ we get the ratio of at least $7/6$. And indeed, the optimal incremental code of size 3 is $\{0^n, 0^{n/7}1^{6n/7}, 1^{4n/7}0^{3n/7}\}$. Indeed, $\mathsf{opt}\text{-}d(2) = n$ and $\mathsf{opt}\text{-}d(3) = 2n/3$ (via $\{0^{2n/3}1^{n/3}, 0^{n/3}1^{n/3}0^{n/3}, 1^{n/3}0^{2n/3}\}$), so both 2- and 3-prefixes have ratios $n/(6n/7) = 7/6$ and $(2n/3)/(4n/7) = 7/6$.*