Error-Correcting Codes for Automatic Control

Rafail Ostrovsky, Yuval Rabani, and Leonard J. Schulman

***Abstract*—Systems with automatic feedback control may consist of several remote devices, connected only by unreliable communication channels. It is necessary in these conditions to have a method for accurate, real-time state estimation in the presence of channel noise. This problem is addressed, for the case of polynomial-growth-rate state spaces, through a new type of error-correcting code that is online and computationally efficient. This solution establishes a constructive analog, for some applications in estimation and control, of the Shannon coding theorem.**

***Index Terms*—Control theory, Lovász local lemma, state estimation, tree codes.**

I. Introduction

A. Motivation

In many automatic control applications, a device (an engine, a terrestrial or aerial mobile robot, a sensor, etc.) communicates with a base station that controls its actions. The communication may be wireless or wired, synchronous or packet-based. Typically the devices have a limited set of commands/controls/actions/moves that they can execute. Actions by the devices combine with environmental disturbances, to cause a change in the parameters describing the state of the system (such as position or temperature). Such devices need to communicate with the base station regarding their current state and get further instructions. Examples are numerous, and include remote mobility issues (such as space or submarine exploration) and web-based on-line control (such as camera and sensor distributed control) [16], [6].

If the controller is physically remote from the sensors or actuators, information flow between them can be subject to noise; if so, system performance depends upon encoding the transmissions against channel noise. The objective of the base station is to learn as precisely as possible the current state of each device in its parameter space. The encoding of communications against channel noise faces a special difficulty in control (as compared to more conventional communications) due to the need for real-time response to transmissions, *causal* encoding (bits of the code can depend only on past events), and causal decoding.

Naturally, in a bounded-capacity channel, there is a tradeoff between (on the one hand) the accuracy and reliability of the information known at the base station, and (on the other) the delay allowed for the communication. It is therefore a challenge to perform channel coding subject to a channel capacity constraint.

The problem can be considered within a very general framework of interactive communication problems [22]; however, the best results in that literature, while establishing the existence of causal codes, remain nonconstructive. Fortunately, there is a feature of some control applications that makes them easier than general interactive-communication problems: controlled devices can often be described with a finite-dimensional parameter space. (Example: the location, orientation, velocity and engine RPM of an aerial drone.) Crucially for the present paper, in such a parameter (or state) space the growth rate of the state space around any point is polynomially bounded.

At each step in time the remote device may send a constant number of transmissions to the base station to update its position/configuration in state space. The objective of the base station is to determine as accurately as possible, despite channel noise, the position of the device in its state space. Of course, one cannot ask that the base station already have high certainty about the real value of any bit which the remote device is trying to transmit, before a significant number of subsequent channel characters have been received. More specifically, if the channel has bounded-size input and output alphabets, is memoryless, and has nonzero probability for every input-output transition, then the best result one can look for is that for a device state s_{wrong} which is *not* the correct current state s_{correct}, if all viable device histories (paths in the state space) leading to s_{wrong} diverge from the true history at least n time steps previously, then the base station should have probability $\exp(-\Omega(n))$ of incorrectly estimating s_{wrong} as the current state of the device. The meaningful question is: Can we achieve such a bound? Doing so demands that encoded characters convey information about events arbitrarily far in the past, because if a message bit (i.e., some datum about about the state of the transmitter) ever stops affecting channel transmissions, then the probability of error in decoding that datum cannot be further reduced.

In this paper, we achieve the desired $\exp(-\Omega(n))$ error probability profile; both our encoding and decoding are constructive and efficient. (Here n is as above, G is a finite-dimensional grid, and the capacity required of the communication channel does not grow with n.) No other method is known of achieving this

Manuscript received August 30, 2006; revised June 16, 2008. Current version published June 24, 2009. The work of R. Ostrovsky was supported in part by the Institute for Pure and Applied Mathematics (IPAM); in part by a gift from Teradata, Intel equipment Grant, IBM Faculty Award, Xerox Innovation Group Award, National Science Foundation under Grants 0430254, 0716835, 0716389, 0830803, a U.C. MICRO Grant, and Okawa Foundation. The work of Y. Rabani was supported by the Israel Science Foundation under Grant 52/03 and by United States–Israel Binational Science Foundation under Grant 2002282. This work was also supported in part while Y. Rabani was visiting the Institute for Pure and Applied Mathematics in the University of California at Los Angeles. The work of L. J. Schulman was supported by the National Science Foundatio (NSF) under Grants CCF-0515342, NSA H98230-06-1-0074, NSF under Grant ITR CCR-0326554, and the Okawa Foundation. The material in this paper was presented in part at the 46th Annual Symposium on Foundations of Computer Science (FOCS), Pittsburgh, PA, October 2005.

R. Ostrovsky is with the Computer Science Department and Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095 USA (e-mail: rafail@cs.ucla.edu).

Y. Rabani is with the Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: rabani@cs.technion.ac.il).

L. J. Schulman is with the California Institute of Technology, Pasadena, CA 91125 USA (e-mail: schulman@caltech.edu).

Communicated by V. A. Vaishampayam, Associate Editor At Large.

Digital Object Identifier 10.1109/TIT.2009.2021303

goal except nonconstructively (through the existence proofs for codes given in [23], [24]).

B. Problem Statement and Results

In this paper we initiate the study of causal error-correcting codes for continual communication of the state of a device in a finite-dimensional parameter space. We restrict ourselves to a simple communication system having one transmitter and one receiver, connected by a discrete memoryless channel without feedback. Our results are fairly insensitive to further details of the channel; for simplicity in what follows we suppose only that the channel has binary input, binary output, and positive capacity. (Our results would go through for any discrete memoryless channel with constant size input and output alphabets, and positive capacity.) The state of the transmitter at any time t is identified with a vertex (which we denote x_t) of a state graph (which we denote G); the graph (which may be directed or undirected and will typically have self-loops) is known to both parties, as is the initial state x_0 of the transmitter. (A single initialization transmission may be required to justify the last assumption, but this is a conventional message-transmission problem, easily solved with a block code.) Let Δ denote the greatest out-degree or in-degree in G.

In each round, the state of the transmitter shifts to an out-neighbor of the previous state. The transmitter can then use the channel a fixed number of times M; the transmitted characters can depend upon the entire history of the transmitter. Our problem is to design efficient encoding and decoding methods which enable the receiver to perpetually maintain estimates of the current transmitter state that are accurate with high probability. We say that the *rate* of the code is $\rho = (\log \Delta)/M$.

The receiver, based upon the transmissions it has received up through time t, has at time t a guess $\hat{x}_t$ of the current state of the transmitter. (We use the term communication scheme to include both the code and the estimation procedure used by the receiver.) For nodes $x, x' \in G$ let $d_G(x, x')$ be the length (number of edges) of a shortest path from x to x' in G. We say that the communication scheme has *error exponent* κ if $P(d_G(x_t, \hat{x}_t) \geq \ell) \leq \exp(-\kappa\ell)\, \forall t, \ell$. We say that the communication scheme is *online-efficient* if the time and space complexities of encoding and (in expectation) decoding are $(\log t)^{O(1)}$. Finally, we say that the communication scheme is *asymptotically good* if it has positive rate, positive error exponent and is online-efficient.

Let $B(x, \ell) = \{x' : d_G(x, x') \leq \ell\}$. The growth of G as a function of ℓ is the supremum over all x of $|B(x, \ell)|$. If this is bounded above by a polynomial in ℓ we say G has polynomial growth. Finite-dimensional grids, which are the graphs for which we construct effective codes in this paper, have polynomial growth. (Throughout the paper, constants implicit in big-O -notation can depend on properties of the noisy channel and on the growth rate of G, i.e., in the constructive case, the dimension of the grid; but not on any other parameters such as the size of the graph or the elapsed time in the protocol.)

C. Our Work

Our main result is the construction of an asymptotically good communication scheme for finite-dimensional grid graphs. The method extends to other graphs that are discretizations of finite-dimensional manifolds, which is typically what is needed in a control application. For example, toroidal meshes. But we do not spell out these variations. (The existence proof for asymptotically good codes, defined below, also holds for arbitrary state graphs G.)

D. A Caveat on Short-Time Dynamics

"Polynomial growth graphs" are not a good model in the regime of very short time-scale dynamics of an unstable system. Consider a system of even the simplest form: linear dynamics in $\mathbb{R}^d, x_{t+1} = Ax_t + Bu_t$, with $||A||_{\text{op}} > 1$. (Following standard notation, x_t is the position at time t, u_t is the actuation impulse at time t, and A and B are real matrices.) In the absence of restorative actuations, and if the state must be communicated just as precisely far from the origin as when near it, then the system in this model can access geometrically increasing volumes of space in successive time steps. At a fixed precision level this means that the number of discretized states accessible to the system grows exponentially in time. Our model is motivated by the larger time scales at which, due either to the true nonlinear system dynamics, or to the implementation of local control, or to a willingness to communicate state less accurately far from the origin, the number of accessible discretized states grows only polynomially in time. The prototypical example is a locally stabilized system that nonetheless drifts over time in $\mathbb{R}^d$ (either absolutely or relative to a reference path), such as a robot with autonomous locomotion.

E. Comment on Asymptotic Goodness

A vast literature on block codes (and on convolutional codes of small constraint length) is devoted to the design of codes whose rate is not merely positive, but approaches closely the capacity of the underlying noisy channel. In the online coding problem that we consider such a goal is rather distant: the present paper is devoted solely to the qualitative establishment of a computationally efficient code of positive rate and positive error exponent. For this reason even our definition of asymptotic goodness glosses over the distinctions between different channels.

We do not minimize the importance of designing near-optimal codes for realistic channel models, and such optimization will be necessary in order to convert any of our ideas into practice. The rationale for our free treatment of constants is twofold. 1) Optimization should be performed in the context of a particular class of channels. Our ideas can be adapted to many channels, but the optimization needed will vary. 2) Unlike in the classical coding literature, it is entirely unclear in our scenario what is the greatest possible achievable rate. It is most unlikely that heuristic optimizations would approach that unknown target.

To expand on the last point: It is clear that M must be at least proportional to $\log \Delta$ in order to transmit with low probability of error the very latest transition in the state space—and our result shows that M needn't be more than a constant factor greater than this—but, in contrast to the classical Shannon theory, it is quite possible that the optimal value of M is more than a factor 1

greater than the obvious lower bound. In general, not enough is yet understood about the rate implications of delay constraints, in spite of important contributions such as [11].

F. Comment on the Online-Efficiency Requirement

The methods in this paper are deterministic, and the expectation in the definition is taken with respect to channel noise. The probability distribution in our results has an exponential tail. More generally one could allow randomized encoding and decoding methods, especially if this led to optimization of the communication rate. Because of the real-time constraint one should however continue to insist upon obtaining light-tailed distributions. (This requirement can be formalized in various ways; a good one is convergence of all finite moments.)

G. Previous Work

Existing error-correcting codes do not provide a satisfactory solution for automatic control applications, as we now explain. Existing error-correcting codes fall mainly into two classes: block codes and convolutional codes. In a block code (with block-length, say, k), a time-stream of data is broken into segments of length k; after an entire segment arrives at the encoder, it is transformed into a (somewhat longer) sequence of bits, which are then sent across the channel.

With block codes it is possible to achieve very low probabilities of error (exponentially small in k) with modest computational load (near-linear in k); however, there is a built-in delay of k time units. This violates the real-time performance requirement of an automatic control application.

Convolutional codes [33], [20], [4] avoid the delay drawback of block codes by performing causal or "on-line" encoding, in which each bit of the input stream immediately starts influencing the encoded message bits, and continues to do so until the end of a time interval of length k, called the constraint length of the code; this interval, which in existing implementations is finite, is analogous to the block length of a block code. The decoder can make an informed guess about a message bit very shortly after its arrival at the encoder, and this guess can continue to be updated during the entire constraint length, with error probability decreasing ultimately to a value exponentially small in k. Although this is the kind of code we would like to use for control, the reason that existing convolutional codes cannot be used is that no efficient constructions are known for convolutional codes with large constraint lengths (unlike the situation for block codes). Indeed, while convolutional codes are heavily used in practice (e.g., in mobile phones), it is thanks to their very short constraint lengths that they have been intensively optimized. The not-very-low probability of error that is a corollary of short constraint length is sufficient for an application in which error events simply cause audio static or drops; however, it is not adequate for control applications in which system stability and performance depends upon preventing accumulation of errors over extended time periods.

Convolutional codes with long, and even infinite, constraint lengths do exist—but not in a form that we can use. The very first papers on convolutional codes show that randomized families of convolutional codes have attractive properties; however, such a family cannot be used without the crutch of a supply of shared random bits at encoder and decoder. More recently, a class of explicit "tree codes" was introduced, which eliminates the need for shared (or even private) coins [23], [24]. (We describe more exactly what these codes are below.) However, while these codes have been shown to exist, the existence proof has not yet been matched by an effective construction, and for that reason, these codes too cannot yet be used. (A similar situation persisted for block codes after Shannon's existence proof for block codes [26] until explicit constructions were provided [7], [5].)

There has recently been substantial progress in information-theoretic and rate-distortion bounds for control applications [31], [17], [18], [32], [29], [14], [3], [21], [15], [8], [34], [27], [28], [11], [2], [10], [13]; some of the roots of these investigations even go back much further [30], [9], [12]. In particular, Sahai's work on "anytime information theory" [21] is related to ours in being concerned with delay-sensitive communication, but unrelated in that the information to be communicated is generated at a positive entropy rate and all of it must ultimately be decoded correctly (in contradistinction to our notion of "trajectory code"). Under these conditions Sahai can (among other results) obtain tight characterizations of the "anytime capacity" (capacity subject to decoding within finite mean-squared time) of various canonical channels (e.g., erasure with feedback, AWGN with feedback). In short, all these works solve different problems than the one considered here. There does not appear to be a prior code for our problem that is efficient in both computation and communication.

Our work, therefore, should be understood as introducing a new family of convolutional codes with infinite constraint length, suitable specifically to control applications but not to general-purpose communication, and which manages to thereby avoid the technical difficulties that have prevented effective construction of general-purpose convolutional codes with infinite (or even long) constraint length.

II. Trajectory Codes

Throughout, G is a graph with vertex set V, initial vertex $x_0 \in V$, and edge set $E \subseteq V \times V$. A trajectory γ of length $|\gamma| = t$ and which begins at time t_0 is a mapping from $\{t_0, \ldots, t_0 + t\}$ to V for which all $(\gamma(i), \gamma(i+1)) \in E$. If two trajectories γ, γ' are of equal length, start at the same time t_0, and share the same start vertex (i.e., $\gamma(t_0) = \gamma'(t_0)$), we write $\gamma \sim \gamma'$. The distance τ between trajectories $\gamma \sim \gamma'$ of length t is $\tau(\gamma, \gamma') = |\{i : t_0 < i \leq t_0 + t \text{ and } \gamma(i) \neq \gamma'(i)\}|$.

A *trajectory code* using an alphabet S is a mapping $\chi : V \times \mathbb{N} \to S$ (where $\mathbb{N} = \{1, 2, \ldots\}$), extended to a mapping from trajectories to S^* by concatenation: $\chi(\gamma) = (\chi(\gamma(t_0 + 1), t_0 + 1), \ldots, \chi(\gamma(t_0 + t), t_0 + t))$. Hamming distance between equal-length words in S^* is denoted h. The *relative distance* of the code is defined to be $\delta = \inf_{\gamma \sim \gamma'} \{h(\chi(\gamma), \chi(\gamma'))/\tau(\gamma, \gamma')\}$. A finite-time trajectory code is defined similarly by a mapping $\chi : V \times \{1, 2, \ldots, T\} \to S$.

The reader should note that this definition is very restrictive. The encoding at time t is allowed to depend only on t and on the

current state (vertex in G), *not* in any other way on the previous history. There is no *a priori* reason to bar codes which depend upon the entire history. The reason we "tie our hands" is that this strategy enables us to find an effective construction of trajectory codes, circumventing the still-open problem of effectively constructing tree codes. (And see the end of this section for more on this subject.) Thanks to the restrictive definition of trajectory codes, our code design will be able to exploit the polynomial growth rate of the space $V \times \{1, 2, \ldots, T\}$, instead of having to cope with the the exponential growth rate of the message stream $\Delta^{\mathbb{N}}$ that corresponds to all trajectories.

We say that a code is *asymptotically good* if it has both positive rate ρ and positive relative distance δ. The next lemma shows that an asymptotically good code is the crucial component of an asymptotically good communication scheme.

(A word on our use of the alphabet S above. Although our underlying physical noisy channel is assumed to be binary, we of course use this channel multiple times per round of the protocol—this is unavoidable, if only because Δ may be larger than two—and we need to control this number of uses, which is inversely proportional to the rate we achieve. In the rest of this paper it will frequently be convenient to speak of coding with a large finite alphabet S. It will be implicit that we use a concatenated code scheme, whereby the characters of S are encoded into binary strings of length $O(\log S)$; the constant in this "O" will depend on the underlying noisy channel and on the desired error exponent κ.)

Lemma 1: If the t'th character of an asymptotically good code with alphabet S can be computed in time and space $(\log t)^{O(1)}$ then the code can be used to construct an asymptotically good communication scheme.

Proof: The code conversion is by simple repetition (the alphabet of the new code is S^k for constant k), and serves only to improve the error exponent. Decoding is by minimum-distance. The expected time and space of the computation is $(\log t)^{O(1)}$ because for sufficiently large k (the transition point for this argument is around $\log \Delta$), the error exponent is large enough that the minimum-distance decoding is unlikely to need to even examine trajectories that diverge far in the past from the trajectory decoded in the previous round. (The argument is similar to one in [24].) The detailed explanation follows.

The decoding algorithm maintains at all times t a trajectory γ^t such that $\chi(\gamma^t)$ has minimum Hamming distance to the received sequence (call it $y(1), \ldots, y(t)$), among all trajectories which start at x_0 at time 0 and end at time t.

The property that ensures that γ^t can usually be updated quickly on the basis of γ^{t-1} is this: let γ be a trajectory and let $1 \leq s \leq t$. If $h(\chi(\gamma(r), \ldots, \gamma(t)), (y(r), \ldots, y(t))) < (t + 1 - r)\delta/2$ for every $1 \leq r \leq s$, then $\gamma(s)$ is on γ^t.

In order to compute γ^t we find the greatest s with the above property on γ^{t-1}, and choose the best among all extensions of $\gamma^{t-1}(s)$. The runtime for doing this is proportional to Δ^{t-s}. Note, $t - s$ is bounded by the greatest suffix of $y(1), \ldots, y(t)$ in which the proportion of channel errors is at least $\delta/2$. Due to memorylessness of the channel, the fraction of channel errors in a suffix has an exponential tail bound; by choosing k sufficiently large, the base of this tail bound can be made less than $1/\Delta$, ensuring in turn that the computation time for minimum-distance decoding is constant in expectation (as well as having an exponential tail bound). □

Our task therefore is to construct an asymptotically good trajectory code. The first problem is to show that such codes exist (Section III). Interestingly, the only proof we know is non-constructive; however, with the aid of this proof we provide a constructive and online-efficient finite-time code for grids. (Section IV).

Comparison With Tree-Codes: It is instructive to compare the present work with that on (explicit) tree codes. These codes were developed in [23], [24], and they are a special case of what we here call trajectory codes; the role of the graph G (in the current notation) is played, in that work, by the protocol tree used by the parties to solve the communication problem if they have access to a noiseless channel. The tree code used in that work for a noisy-communication protocol is a particular case of what we now call the trajectory code on $V \times \mathbb{N}$. The existence proof provided in the earlier work relies on the tree structure of the graph, and does not apply to the more general case considered here. However, the purpose of the generalization is not to handle more difficult communication problems; the case that G is a tree is actually the most difficult one. (Using tree codes enables eventual reconstruction of the entire history of the transmitter, not only reconstruction of a good estimate of the current state.) Instead, the purpose in our paper is to obtain a computationally effective solution using the special assumption that G has polynomial growth. (As discussed earlier, this assumption is motivated by control applications, with G being a discretization of the finite-dimensional parameter space of the system.) Thus, we circumvent the open problem of explicitly constructing a tree code. The only progress we are aware of regarding tree codes is an existence proof for codes of improved rate, by Peczarski [19]; to our knowledge there has been no progress toward an effective construction. For some minor notes on the topic the reader is also referred to [25].

Here is another way to think about our work: a naïve approach to the code design problem takes advantage of conventional source-channel separation and views the message "source" as a stream of edge-moves, i.e., an element of the set $\Delta^{\mathbb{N}}$, where $\mathbb{N}$ represents the natural numbers. But this approach encounters a technical obstacle (construction of tree codes). Instead our definition of trajectory codes incorporates the desired semantics of our application: namely, it does not matter if the receiver (base station) has a misconception about some portion of the past history of the transmitter, so long as the receiver is up-to-date on its current position.

III. Existence of Asymptotically Good Trajectory Codes

Theorem 2: Every graph G possesses an asymptotically good trajectory code. Furthermore, every $\delta < 1$ is feasible as the relative distance of an asymptotically good code.

Proof: In order to achieve positive rate our code $\chi : V \times \mathbb{N} \to S$ must use an alphabet S of size $\Delta^{O(1)}$. Consider using a random χ, i.e., one in which each label is selected independently and uniformly. A code obtained in this manner is almost-surely

not asymptotically good. In fact, with probability 1 there will be infinitely many pairs of distinct trajectories $\gamma \sim \gamma'$ which are labeled identically, i.e., for which $h(\chi(\gamma), \chi(\gamma')) = 0$. (Think just of trajectories of length 1.) Nonetheless we can use this probability space over codes to show the existence of the desired code.

Consider first the finite-graph, finite-time restriction of the problem to $B(x_0, T) \times \{1, 2, \ldots, T\}$. Fix any desired relative distance bound δ. If $\gamma = (\gamma_1, \gamma_2)$ consists of two trajectories such that $\gamma_1 \sim \gamma_2$ and which share only their common start vertex (i.e., $\tau(\gamma_1, \gamma_2) = |\gamma_1|$), then we refer to γ as a pair of "twins" and write $|\gamma| = |\gamma_1|$ and $h\chi(\gamma) = h(\chi(\gamma_1), \chi(\gamma_2))$. Note that $\inf_{\gamma_1 \sim \gamma_2}(h(\chi(\gamma_1), \chi(\gamma_2))/\tau(\gamma_1, \gamma_2)) = \inf_{\text{twins}\gamma}(h\chi(\gamma)/|\gamma|)$. For a pair of twins γ let A_γ be the event that $h\chi(\gamma)/|\gamma| < \delta$. Due to the Chernoff bound for large deviations, there is (for any δ) a positive c for which $P(A_\gamma) \leq |S|^{-c|\gamma|}$ for all γ. The code achieves relative distance δ if $\bigcap \overline{A_\gamma} \neq \emptyset$.

For twins $\gamma = (\gamma_1, \gamma_2)$ let $N_\gamma = \{\text{twins}\beta = (\beta_1, \beta_2) : \exists \epsilon_1, \epsilon_2 \in \{1, 2\}, j_1, j_2 > 0$ such that $\gamma_{\epsilon_1}(j_1) = \beta_{\epsilon_2}(j_2)\}$. Observe that A_γ is independent of the random variable $(A_\beta)_{\beta \notin N_\gamma}$.

The Lovász local lemma [1] ensures that $\bigcap \overline{A_\gamma} \neq \emptyset$ provided that there exist nonnegative reals $0 \leq x_\gamma < 1$ for which

$$x_\gamma \prod_{\beta \in N_\gamma} (1 - x_\beta) \geq P(A_\gamma).$$

Observe that $|\{\beta : \beta \in N_\gamma, |\beta| = \ell\}| \leq 4|\gamma|\ell\Delta^{2\ell}$. (Up to this point we have used only the assumption that Δ is an upper bound on out-degrees; here we also employ the assumption that it is an upper bound on in-degrees.) Let c' be sufficiently large so that for $\Delta \geq 2$, (a) $1 - \Delta^{-c'\ell} \geq e^{-2\Delta^{-c'\ell}}$; (b) $\sum_{\ell=1}^{\infty} \ell\Delta^{(2-c')\ell} \leq 2$. Set $x_\gamma = \Delta^{-c'|\gamma|}$. Then

$$\begin{aligned} x_\gamma \prod_{\beta \in N_\gamma} (1 - x_\beta) &\geq \Delta^{-c'|\gamma|} \prod_{\ell=1}^{\infty} (1 - \Delta^{-c'\ell})^{4|\gamma|\ell\Delta^{2\ell}} \\ &\geq \Delta^{-c'|\gamma|} \prod_{\ell=1}^{\infty} e^{-8\Delta^{-c'\ell}|\gamma|\ell\Delta^{2\ell}} \\ &= \Delta^{-c'|\gamma|} e^{-8|\gamma| \sum_{\ell=1}^{\infty} \ell\Delta^{(2-c')\ell}} \\ &\geq \Delta^{-c'|\gamma|} e^{-16|\gamma|}. \end{aligned}$$

Since $P(A_\gamma) \leq |S|^{-c|\gamma|}$, the hypotheses of the local lemma are met with an alphabet of size $\Delta^{O(1)}$.

To extend the proof to the general case we apply a standard compactness argument (see [1]). For any T, the trajectory codes on $B(x_0, T) \times \{1, 2, \ldots, T\}$ ensured by the above argument form a finite nonempty set. Let C_T denote the set of codes on $V \times \mathbb{N}$ which restrict to one of the trajectory codes on $B(x_0, T) \times \{1, 2, \ldots, T\}$. C_T is a nonempty set that is closed in the product topology on $S^{V \times \mathbb{N}}$. Note that $C_T \subseteq C_{T-1}$; the intersection of the sets C_T for any finite number of indices T is therefore nonempty. The set $\bigcap_{t \in \mathbb{N}} C_T$ is the desired set of trajectory codes with relative distance δ. By Tychonoff's Theorem, $S^{V \times \mathbb{N}}$ is compact. Therefore $\bigcap_{T \in \mathbb{N}} C_T \neq \emptyset$. □

IV. Construction of Trajectory Codes for Grids

We now construct, for a grid graph of any finite dimension d and for any desired relative distance $\delta < 1$, an asymptotically good online-efficient finite-time trajectory code. We actually provide two different constructions: the first, described in Section IV-A, is the simpler of the two. Its drawback is that the rate of the code scales in the dimension as $\exp(-d)$, whereas the only upper bound we are aware of is $O(1/d)$. (This upper bound is straightforward: $2d$ is the degree of the d-dimensional grid graph. The upper bound derives therefore from the requirement to separate twin trajectories of length 1, in other words, from the very short time-delay requirements on the code.) The second construction, described in Section IV-B, almost closes the gap by achieving rate $\Theta(1/d^3)$ (and in a somewhat less efficient offline construction, $\Theta(1/d^2)$).

Let G be the graph on vertex set $V_{n,d} = \{-n/2 + 1, \ldots, n/2\}^d$ with an edge from $(u_1, \ldots, u_d)$ to $(v_1, \ldots, v_d)$ if $|u_i - v_i| \leq 1$ for all i. For simplicity we describe the construction for a time bound of $n/2$. So our task is to construct a trajectory code $\chi : V_{n,d} \times \{1, \ldots, n/2\} \to S$ of relative distance δ.

A. Construction I: "Multiple Overlaid Tilings"

The idea is to combine recursion with use of an explicit block code. Set $n_1 \in \Theta(\log n)$. (n_1 needs only to be large enough to accommodate codewords of the block code described below.) Let k be the least even integer greater or equal to $\frac{12}{1-\delta} + 4$. For simplicity assume that kn_1 divides n.

1) Recursive Construction: The block code: Let $\eta : V_{n,d} \to R_1^{n_1}$ (for a finite alphabet R_1) be an asymptotically good block code (i.e., one with positive rate and positive relative distance) of relative distance $(1 + \delta)/2$, in which encoding and decoding can be performed in time $n_1^{O(1)}$. Rewrite η as a mapping $\eta_1 : V_{n,d} \times \{1, \ldots, n_1\} \to R_1$, so that for $x \in V_{n,d}$, $\eta(x) = (\eta_1(x, 1), \ldots, \eta_1(x, n_1))$.

The recursive code: Let $\chi_1 : V_{kn_1,d} \times \{1, \ldots, kn_1/2\} \to S_1$ (for a finite alphabet S_1) be a trajectory code of relative distance $(1 + \delta)/2$.

The basic idea is to cover $V_{n,d} \times \{1, \ldots, n/2\}$ by overlapping tiles. Each tile is "placed" at a specified $x \in V_{n,d} \times \{0, \ldots, n/2 - 1\}$, and is the following mapping:

$$\begin{aligned} \sigma_x : &\left(\prod_{i=1}^{d} \{x_i - kn_1/2 + 1, \ldots, x_i + kn_1/2\} \right) \\ &\qquad \times (x_{d+1} + 1, \ldots, x_{d+1} + kn_1/2) \to S_1 \times R_1 \\ \sigma_x(y) = &(\chi_1(y - x), \eta_1(x_1, \ldots, x_d, (y_{d+1} - x_{d+1} \bmod n_1))). \end{aligned}$$

The cover of $V_{n,d} \times \{1, \ldots, n/2\}$ by overlapping tiles will be described by a union of several covers, each of which is a tiling (a cover by non-overlapping tiles). Each tiling is associated with a vector $(a_1, \ldots, a_{d+1}) \in \{-k/2 + 1, \ldots, k/2\}^d \times \{0, \ldots, k - 1\}$. (Strictly speaking each tiling may fail to be a cover but only due to edge effects which we gloss over.) The collection of tiles associated with the label $(a_1, \ldots, a_{d+1})$ consists of those placed at x of the form

$$x = n_1(kz_1 + a_1, \ldots, kz_{d+1} + a_{d+1})$$

for all $(z_1,\ldots,z_{d+1})$ of the form

$$(z_1,\ldots,z_{d+1}) \in \{-n/(2kn_1)+1,\ldots,n/(2kn_1)\}^d \times \{1,\ldots,n/(2kn_1)\}$$

The tiling labeled $(a_1,\ldots,a_{d+1})$ therefore defines a mapping

$$\chi_{a_1,\ldots,a_{d+1}} : V_{n,d} \times \{1,\ldots,n/2\} \to S_1 \times R_1$$

by restriction (except possibly near the boundaries due to fencepost errors).

The trajectory code χ is the concatenation of the codes associated with each of the tilings

$$\chi(y) = \left(\chi_{a_1,\ldots,a_{d+1}}(y)\right)_{a_1,\ldots,a_{d+1}}$$

Observe that the number of labels concatenated at each vertex is k^{d+1}.

Lemma 3: χ achieves relative distance δ.

Proof: Consider any twins (γ,γ'). Let $t = |\gamma|$ and let t_0 be the starting time of the pair of trajectories.

If $t \le (k-4)n_1/2$ then the pair (γ,γ') is contained entirely within a tile. This implies relative distance at least $(1+\delta)/2$.

Otherwise, partition the time period $[t_0, t_0+t]$ into consecutive blocks of the following lengths:

$$\ell_1, m_1, \ell_2, m_2, \ldots, \ell_{L-1}, m_{L-1}, \ell_L$$

(for L to be determined), by the following procedure. (Define $t_i = t_0 + \sum_{j=1}^{i}(\ell_j + m_j)$.)

Suppose $\ell_1, m_1, \ldots, \ell_{i-1}, m_{i-1}$ have already been defined. Set $\ell_i = \min\{t+t_0-t_{i-1}, (k-4)n_1/2\}$. If $\ell_i = t+t_0-t_{i-1}$, set $L=i$ and halt. (It may happen that $\ell_i = 0$ but only if $L=i$.) Otherwise set m_i to be $t+t_0-\ell_i-t_{i-1}$ if the following set is empty, and otherwise to be its least element:

$$\{m \ge 0 : d_G(\gamma(t_{i-1}+\ell_i+m),\ \gamma'(t_{i-1}+\ell_i+m)) \le 2n_1\}.$$

(It may happen that $m_i = 0$.)

Since $t > (k-4)n_1/2, L \ge 2$. Observe that for each i, $\ell_i = (k-4)n_1/2$, except that ℓ_L may be smaller.

We show that within each of the blocks, the Hamming distance between the γ and γ' codewords is at least $-n_1 + (1+\delta)\ell_i/2$ or $-n_1+(1+\delta)m_i/2$, as the case may be.

We begin with the "m_i type" blocks. For the duration of such a block, the trajectories are separated by graph distance at least $2n_1$. In each time segment of length n_1, aligned with the tiles of the construction, the two trajectories pass through distinct codewords of η, and experience relative distance $(1+\delta)/2$. The first and last time segments can be incomplete and therefore less efficient, but the total number of shared characters due to these two time segments is bounded by $(1-\delta)n_1$, which we upper bound by n_1.

Next we treat the "ℓ_i type" blocks, with the following "virtual trajectory" argument. Choose a vertex $y = (y_1,\ldots,y_d) \in V_{n,d}$ such that both $d_G(y,\gamma(t_{i-1})) \le n_1$ and $d_G(y,\gamma'(t_{i-1})) \le n_1$. Define $\tilde{y} \in V_{n,d} \times \{1,\ldots,n/2\}$ by $\tilde{y} = (y_1,\ldots,y_d,t_{i-1}-n_1)$. Construct a trajectory $\tilde{\gamma}$ with start time $t_{i-1}-n_1$ and length ℓ_i+n_1 by having it start at $\tilde{\gamma}(t_{i-1}-n_1) = \tilde{y}$, reach $\tilde{\gamma}(t_{i-1}) = \gamma(t_{i-1})$, and thereafter be identical to γ until time $t_{i-1}+\ell_i$. Similarly construct a disjoint trajectory $\tilde{\gamma}'$ with start time $t_{i-1}-n_1$ and length ℓ_i+n_1 which starts at $\tilde{\gamma}'(t_{i-1}-n_1) = \tilde{y}$, reaches $\tilde{\gamma}'(t_{i-1}) = \gamma(t_{i-1})'$, and thereafter is identical to γ' until time $t_{i-1}+\ell_i$. Observe that $\tilde{\gamma}$ and $\tilde{\gamma}'$ are twins of length at most $(k-2)n_1/2$, so there is a tile entirely containing them. Hence the Hamming distance between their words is at least $(\ell_i+n_1)(1+\delta)/2$, and therefore the Hamming distance between the segments of γ and γ' is at least $-n_1+(\ell_i+n_1)(1+\delta)/2 \ge -n_1+\ell_i(1+\delta)/2$.

Combining the contributions of all time segments, we find that the Hamming distance between the two words is at least $-(2L-1)n_1+(1+\delta)t/2$. Note that $t \ge (L-1)(k-4)n_1/2$. Recalling that $n_1 < 2t/(k-4)$, this implies that $(2L-1)n_1 < 6t/(k-4)$. Hence the Hamming distance is greater than $t(\frac{1+\delta}{2} - \frac{6}{k-4}) \ge t\delta$. □

2) The Code: What is left unstated by the above construction, is how the code χ_1 on the tiles is constructed. The two extreme options are to pursue the whole construction recursively, or to construct χ_1 by exhaustive search. The former option is unsatisfactory because of the alphabet blow-up at each level of recursion. The latter option requires a one-time offline $\exp((\log n)^{d+1})$-time computation. Once χ_1 has been constructed, local look-up can be performed in time $\log^{O(1)} n$, hence achieving online-efficiency. In order to improve the offline efficiency, we implement just one more level of recursion, constructing χ_1 out of a code χ_2 for tiles of size $\log\log n$, which is itself constructed by exhaustive search in time $\exp((\log\log n)^{d+1})$. Thus, using the option of constructing χ_2, we have the following theorem.

Theorem 4: The above "multiple overlaid tilings" construction of χ can be constructed offline in sublinear time for any fixed dimension d, is online-efficient, and achieves any required relative distance $\delta < 1$. Its rate is proportional to $\exp(-d)$.

Proof: The relative distance guarantee follows from Section IV-A; the construction efficiency follows by combining the construction of Section IV-A with the double-recursion of Section IV-B. □

B. Construction II: "Shingles"

As pointed out earlier, the construction of Section IV-A gives a communication rate proportional only to 2^{-d}: this is directly due to the fact that the covering of space by tiles, covers every point with multiplicity 2^d.

We now give a much more economical cover of space. The way in which this cover is used to construct a code is slightly less obvious than for the previous cover—simple concatenation of characters is no longer sufficient—so we defer the code description, and begin with the purely geometric construction.

1) Shingle Covering of Space by Cubes: To keep things abstract we work in $\mathbb{R}^r$ and describe a "shingled" cover of $\mathbb{R}^r$ by tiles that are axis-parallel cubes, each of side-length $1+2^{1-r}$, with the following two properties:

1) Every point is "well inside" some tile: specifically, distance at least 2^{-r} away from its exterior.
2) No point is inside more than $r+1$ tiles.

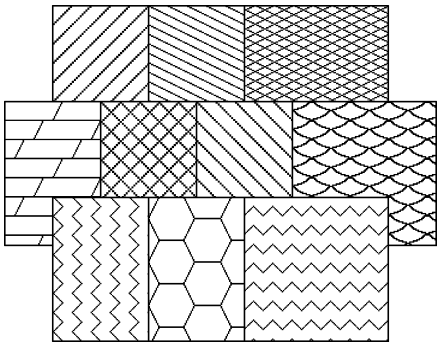

Fig. 1. Shingles in two dimensions.

This is expressed in the following Theorem. Let C_r be the closed unit cube $C_r = [0,1]^r$. Let D_r be a slightly enlarged, open cube: $D_r = (-2^{-r}, 1+2^{-r})^r$.

Theorem 5: There is an (easily computed) cover of $\mathbb{R}^r$ by translations of C_r such that no point is within more than $r+1$ translations of D_r.

Proof: The construction is this. Each integer vector of coordinates $(x_1, \ldots, x_r)$ will index a translate of C_r. By the "least corner" of a cube we mean the corner with the least numerical values in each axis. Place the least corner of the C_r-cube indexed by $(x_1, \ldots, x_r)$ at the point

$$\left(\left(x_1 + \frac{x_2}{2} + \frac{x_3}{4} + \frac{x_4}{8} + \ldots + \frac{x_r}{2^{r-1}}\right), \left(x_2 + \frac{x_3}{2} + \frac{x_4}{4} + \frac{x_5}{8} + \ldots + \frac{x_r}{2^{r-2}}\right), \ldots, \left(x_{r-1} + \frac{x_r}{2}\right), x_r\right).$$

That is to say, construct a tiling by unit cubes by applying the following transformation to integer vectors:

$$\begin{pmatrix} 1 & 1/2 & \ldots & 2^{2-r} & 2^{1-r} \\ & 1 & 1/2 & \ldots & 2^{2-r} \\ & & 1 & 1/2 & \ldots \\ & & & 1 & 1/2 \\ & & & & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ \ldots \\ x_r \end{pmatrix}$$

and place a unit cube with its least corner at each image of a point of the integer lattice. The offsetting of the tiles is the reason for the term "shingles." See Fig. 1.

Fix any point $y = (2^{1-r}y_1, \ldots, 2^{1-r}y_r) \in \mathbb{R}^r$. The question is, how many C_r and D_r cubes can y belong to? We wish to show that the former number is at least 1, and the latter at most $r+1$.

Define $Y = (Y_1, \ldots, Y_r)$ by letting each Y_i be an integer such that $Y_i - y_i \leq 1/2$ (ties broken arbitrarily). Note that if y lies in a particular translate of D_r then so does $2^{1-r}Y$. Moreover, if Y lies in a translate of D_r then it also lies in the corresponding translate of C_r. Hence for an upper bound on the number of D_r-cubes containing a point, it suffices to upper bound, for integer $y_1, \ldots, y_r$, the number of C_r-cubes containing $y = (2^{1-r}y_1, \ldots, 2^{1-r}y_r)$.

We begin with the lower bound for arbitrary points y. The C_r-cube defined by $(x_1, \ldots, x_r)$ contains y if and only if for every $1 \leq i \leq r$, there is a real $0 \leq c_i \leq 2^{r-1}$ such that

$$x_i + 2^{-1}x_{i+1} + \ldots + 2^{i-r}x_r + 2^{1-r}c_i = 2^{1-r}y_i \quad \text{(IV.1)}$$

Spelling this out in long-hand, we are interested in the number of solutions (in integer x_i and in real $0 \leq c_i \leq 2^{r-1}$) to the following system of equations defined by real y:

$$2^{r-1}x_r + c_r = y_r \quad \text{(IV.2)}$$
$$2^{r-1}x_{r-1} + 2^{r-2}x_r + c_{r-1} = y_{r-1} \quad \text{(IV.3)}$$
$$\ldots \quad \text{(IV.4)}$$
$$2^{r-1}x_1 + 2^{r-2}x_2 + \ldots + x_r + c_1 = y_1 \quad \text{(IV.5)}$$

We construct what we call the "primary solution" $X = (X_1, \ldots, X_r)$ to this system of equations. First, using the special case of (IV.1) given in (IV.2), set $X_r = \lfloor y_r 2^{1-r} \rfloor$ and $c_r = y_r - 2^{r-1}X_r$. In general, using (IV.1), we set X_i and c_i using

$$X_i = \left\lfloor 2^{1-r}y_i - \sum_{\ell=1}^{r-i} 2^{-\ell}X_{i+\ell} \right\rfloor \quad \text{and} \quad c_i = y_i - \sum_{\ell=0}^{r-i} 2^{r-1-\ell}X_{i+\ell}. \quad \text{(IV.6)}$$

This demonstrates the lower bound (containment in at least one C_r-cube). We now turn to the upper bound (containment in at most $r+1D_r$-cubes). Recall that for this purpose it is sufficient to bound the number of times a point $y = (2^{1-r}y_1, \ldots, 2^{1-r}y_r)$, with $y_1, \ldots, y_r$ integer, is covered by C_r-cubes.

We construct a tree of all solutions to the (IV.1). Each level of the tree corresponds to one of the variables: the level immediately below the root corresponds to x_r, the level below that to x_{r-1}, and so on; the last level corresponds to x_1. Each path down the tree specifies a single solution to the equations.

Here is how we construct the tree, top-to-bottom. At the root (level $r+1$), no variables have been set, and we wish to identify possible settings for x_r. Examining (IV.2), one child of the root is the primary solution, $X_r = \lfloor y_r 2^{1-r} \rfloor$ and $c_r = y_r - 2^{r-1}X_r$. Moreover, if and only if $y_r 2^{1-r}$ is an integer (or equivalently $c_r = 0$ in the primary solution), there is a second solution: $x_r = \lfloor y_r 2^{1-r} \rfloor - 1$ and $c_r = 2^{r-1}$. Each solution (whether there are one or two) is set to be a child of the root in the tree of solutions.

In general, there are several nodes at the $(i+1)$th level of the tree. At this level we are using (IV.1); at any particular node of this level, the variables $x_r, \ldots, x_{i+1}$ have already been set, and the variables x_i and c_i are to be determined. The pattern is just as we saw for x_r—namely, we can always construct x_i from $x_{i+1}, \ldots, x_r$ in the primary method given in (IV.6). Moreover, if and only if $2^{1-r}y_i - \sum_{\ell=1}^{r-i} 2^{-\ell}x_{i+\ell}$ is an integer (or equivalently $c_i = 0$ in the primary method), there is a second solution: $x_i = \lfloor 2^{1-r}y_i - \sum_{\ell=1}^{r-i} 2^{-\ell}x_{i+\ell} \rfloor - 1$ and $c_i = 2^{r-1}$. In this way the node $(x_r, c_r, \ldots, x_{i+1}, c_{i+1})$ at level $i+1$ is assigned either one or two children at level i. We now argue that at most one

node at level $i+1$ is assigned two children; from this it follows by induction that the tree has at most $r+1$ leaves.

Any solution can be viewed as a variant of the primary solution, described by a vector $\varepsilon = (\varepsilon_r, \ldots, \varepsilon_1) \in \{0,1\}^r$; we denote the variant by X^ε, with the primary solution being denoted $X = X^{(0,\ldots,0)}$:

$$\begin{aligned} X_r^\varepsilon &= \lfloor y_r 2^{1-r} \rfloor - \varepsilon_r \\ &\cdots \\ X_i^\varepsilon &= \left\lfloor 2^{1-r} y_i - \sum_{\ell=1}^{r-i} 2^{-\ell} X_{i+\ell}^\varepsilon \right\rfloor - \varepsilon_i \\ &= \left\lfloor 2^{1-r} y_i - \sum_{\ell=1}^{r-i} 2^{-\ell} X_{i+\ell} + \sum_{\ell=1}^{r-i} 2^{-\ell} \varepsilon_{i+\ell} \right\rfloor - \varepsilon_i \end{aligned}$$

In order that the node $(X_r^\varepsilon, \ldots, X_{i+1}^\varepsilon)$ have two children, the value $2^{1-r} y_i - \sum_{\ell=1}^{r-i} 2^{-\ell} X_{i+\ell} + \sum_{\ell=1}^{r-i} 2^{-\ell} \varepsilon_{i+\ell}$ must be an integer. This can be true of only one of the nodes at level $i+1$ because at each of those nodes the quantity $\sum_{\ell=1}^{r-i} 2^{-\ell} \varepsilon_{i+\ell}$ is a distinct number in the interval $[0,1)$. (Specifically, it is a multiple of 2^{i-r}.) □

A final note that will be useful in the next section: observe that for any point in $\mathbb{R}^r$, each of the D_r-cubes covering it is associated with a distinct binary vector $(x_1 \bmod 2, \ldots, x_r \bmod 2)$.

2) Trajectory Code Based on the Shingle Cover: Similarly to Section IV-A, our task is to construct a trajectory code $\chi : V_{n,d} \times \{1,\ldots,n\} \to S$ of relative distance δ. Let $C = 1 + 3/(1-\delta)$. We suppose that we have already constructed a trajectory code of a smaller size, $\chi_1 : V_{n',d} \times \{1,\ldots,n'\} \to S_1$, for $n' = (1+2^{-d})2^{2d+2} C(d+1)\log n$, and with a better relative distance $(1+\delta)/2$. (S_1 is a finite alphabet.) Note that there is no need to repeat the recursion indefinitely; for best asymptotics we recurse twice (as in the previous section) and construct χ_2 by exhaustive search. We use the shingle construction to cover $V_{n,d} \times \{1,\ldots,n\}$ with copies of χ_1. Thus, every point of $V_{n,d} \times \{1,\ldots,n\}$ is at distance at least $2^{-d-1} 2^{2d+2} C(d+1)\log n = 2^{d+1} C(d+1)\log n$ away from the exterior of one of these shingles.

We also use another set of smaller tiles. These are labeled with $\eta_1 : V_{n'',d} \times \{1,\ldots,n''\} \to R_1$ where $n'' = (1+2^{-d})2^{d+1}(d+1)\log n$ and R_1 is a finite alphabet. We also use the shingle construction to cover $V_{n,d} \times \{1,\ldots,n\}$ with copies of η_1. Thus, every point of $V_{n,d} \times \{1,\ldots,n\}$ is at distance at least $(d+1)\log n$ away from the exterior of one of these smaller shingles.

Similarly to the previous construction, η_1 is based upon an asymptotically good block code η of block length $(d+1)\log n$ and relative distance $(1+\delta)/2$, in which encoding and decoding can be performed in time $((d+1)\log n)^{O(1)}$. The full region $V_{n,d} \times \{1,\ldots,n\}$ is covered by approximately $(n/n'')^{d+1}$ of the small tiles. The code η encodes the label of each of these small tiles (i.e., the integer string x which identifies its least corner)—note that the number of bits required for such a label is $O((d+1)\log n)$—and maps it into a block codeword of length $(d+1)\log n$ over the finite alphabet R_1. Let $\eta : V_{n,d} \to R_1^{(d+1)\log n}$ denote this codeword. Then η_1 is defined as follows: in the shingle (of size n'') labeled by $x_1, \ldots, x_{(d+1)\log n}$, and at the coordinate $(u_1, \ldots, u_{d+1})$ within the shingle, η_1 is set to be $\eta_1(u_1,\ldots,u_{d+1}) = \eta(x_1,\ldots,x_{(d+1)\log n})_{(u_{d+1} \bmod ((d+1)\log n))}$. In other words, the codeword identifying the shingle is repeated within the shingle, $(1+2^{-d})2^{d+1}$ times from top to bottom, and without any horizontal variation.

So, every point of the region $V_{n,d} \times \{1,\ldots,n\}$ is covered with between 1 and $d+2$ copies of each of the two types of shingles. The character of χ at each point is the concatenation of two characters which we now specify; each is defined using one of the two types of shingles.

The character defined at a given point by the block-codeword shingles is this. Suppose that the point is contained in shingles $(x_1^1,\ldots,x_{d+1}^1),\ldots,(x_1^D,\ldots,x_{d+1}^D)$ where D (thanks to the previous section) is at most $d+2$. Moreover suppose that the colors (elements of R_1 specified by η_1) assigned to the point by these shingles are $\kappa^1,\ldots,\kappa^D$. If $D < d+2$, pad each of these lists with 0's, namely, set $x_k^j = 0$ and $\kappa^j = 0$ for all $D < j \le d+2$ and for all k. Then, write out the following word as the label of the vertex:

$$\left((x_1^1 \bmod 2, \ldots, x_{d+1}^1 \bmod 2, \kappa^1), \ldots, (x_1^{d+2} \bmod 2, \ldots, x_{d+1}^{d+2} \bmod 2, \kappa^{d+2})\right).$$

(Notice that we do not even care in what order these tuples are written down.)

The idea is that if at the time corresponding to this point, large Hamming distance between the pair of trajectories is being guaranteed by the block code, then the binary (" $\bmod 2$") parts of this character distinguish the two trajectories unless those binary parts line up perfectly; and if they do line up perfectly, the labels of the two distinct shingles are lined up and therefore distinguish the two trajectories.

The character corresponding to the recursive trajectory code is defined in the same way, although the justification for why this works is different, since the recursive code ensures large Hamming distance if the two trajectories pass through the same shingle, not different ones. Just as before, though, the idea is that the " $\bmod 2$" parts of this character distinguish the two trajectories unless those parts line up perfectly; and if they do line up perfectly, the one particular coordinate derived from the shingle containing both trajectories, provides the required Hamming distance.

If the recursive trajectory code χ_1 is constructed by exhaustive search, which can be done in time $\exp((\log n)^{d+1})$, then the number of bits specifying each color of χ is $O(d^2)$. If on the other hand χ_1 is constructed through one more level of recursion, so that χ_2 is constructed by exhaustive search, which can be done in time $\exp((\log\log n)^{d+1})$ (i.e., sublinear), then the number of bits specifying each color of χ (i.e., the number of bits transmitted in each round of communication) is $O(d^3)$.

Lemma 6: χ achieves relative distance δ

Proof: The proof is almost identical to that of Lemma 3. Consider any twins (γ,γ'). Let $t = |\gamma|$ and let t_0 be the starting time of the pair of trajectories.

If $t \le (2^{d+1}C - (1+2^{-d})2^d)(d+1)\log n$ then the pair (γ,γ') is contained entirely within a recursive (i.e., χ_1) tile. This implies relative distance at least $(1+\delta)/2$.

Otherwise, partition the time period $[t_0, t_0+t]$ into consecutive blocks of the following lengths:

$$\ell_1, m_1, \ell_2, m_2, \ldots, \ell_{L-1}, m_{L-1}, \ell_L$$

(for L to be determined), by the following procedure. (Define $t_i = t_0 + \sum_{j=1}^{i}(\ell_j + m_j)$.)

Suppose $\ell_1, m_1, \ldots, \ell_{i-1}, m_{i-1}$ have already been defined. Set $\ell_i = \min\{t+t_0-t_{i-1}, (2^{d+1}C-(1+2^{-d})2^d)(d+1)\log n\}$. If $\ell_i = t+t_0-t_{i-1}$, set $L=i$ and halt. (It may happen that $\ell_i = 0$ but only if $L=i$.) Otherwise set m_i to be $t+t_0-\ell_i-t_{i-1}$ if the following set is empty, and otherwise to be its least element:

$$\{m \geq 0 : d_G(\gamma(t_{i-1}+\ell_i+m), \gamma'(t_{i-1}+\ell_i+m)) \leq n''\}.$$

(It may happen that $m_i = 0$.)

Since $t > (2^{d+1}C - (1+2^{-d})2^d)(d+1)\log n, L \geq 2$. Observe that for each $i, \ell_i = (2^{d+1}C - (1+2^{-d})2^d)(d+1)\log n$, except that ℓ_L may be smaller.

We show that within each of the blocks, the Hamming distance between the γ and γ' codewords is at least $-n''/4+(1+\delta)\ell_i/2$ or $-n''/4+(1+\delta)m_i/2$, as the case may be.

We begin with the "m_i type" blocks. For the duration of such a block, the trajectories are separated by graph distance at least n". In each time segment of length $(d+1)\log n$, aligned with the tiles of the construction, the two trajectories pass through distinct codewords of η, and experience relative distance $(1+\delta)/2$. The first and last time segments can be incomplete and therefore less efficient, but the total number of shared characters due to these two time segments is $\leq (1-\delta)(d+1)\log n \leq (d+1)\log n$, so (using $d \geq 1$) the Hamming distance in the block is $>m_i(1+\delta)/2 - n''/4$.

Next we treat the "ℓ_i type" blocks, with the following "virtual trajectory" argument. Choose a vertex $y = (y_1, \ldots, y_d) \in V_{n,d}$ such that both $d_G(y, \gamma(t_{i-1})) \leq n''/2$ and $d_G(y, \gamma'(t_{i-1})) \leq n''/2$. Define $\tilde{y} \in V_{n,d} \times \{1, \ldots, n/2\}$ by $\tilde{y} = (y_1, \ldots, y_d, t_{i-1} - n''/2)$. Construct a trajectory $\tilde{\gamma}$ with start time $t_{i-1} - n''/2$ and length $\ell_i + n''/2$ by having it start at $\tilde{\gamma}(t_{i-1} - n''/2) = \tilde{y}$, reach $\tilde{\gamma}(t_{i-1}) = \gamma(t_{i-1})$, and thereafter be identical to γ until time $t_{i-1} + \ell_i$. Similarly construct a disjoint trajectory $\tilde{\gamma}'$ with start time $t_{i-1} - n''/2$ and length $\ell_i + n''/2$ which starts at $\tilde{\gamma}'(t_{i-1} - n''/2) = \tilde{y}$, reaches $\tilde{\gamma}'(t_{i-1}) = \gamma(t_{i-1})'$, and thereafter is identical to γ' until time $t_{i-1} + \ell_i$. Observe that $\tilde{\gamma}$ and $\tilde{\gamma}'$ are twins of length $\ell_i + n''/2 \leq (2^{d+1}C - (1+2^{-d})2^d)(d+1)\log n + n''/2 = 2^{d+1}C(d+1)\log n$, so there is a recursive tile entirely containing them. Hence the Hamming distance between their words is at least $(\ell_i + n''/2)(1+\delta)/2$, and therefore the Hamming distance between the segments of γ and γ' is at least $-n''/2 + (\ell_i + n''/2)(1+\delta)/2 \geq \ell_i(1+\delta)/2 - n''/4$.

Combining the contributions of all time segments, we find that the Hamming distance between the two words is at least $-(2L-1)n''/4 + (1+\delta)t/2$. Since $\ell_i = (2^{d+1}C - (1+2^{-d})2^d)(d+1)\log n$ for all $i < L, t \geq (L-1)(2^{d+1}C - (1+2^{-d})2^d)(d+1)\log n$. Hence the Hamming distance is at least

$$\begin{aligned} &t\delta + t(1-\delta)/2 - (2L-1)n''/4 \\ &\quad \geq t\delta + (L-1)(2^{d+1}C \\ &\qquad - (1+2^{-d})2^d)(d+1)(\log n)(1-\delta)/2 \\ &\qquad - (2L-1)(1+2^{-d})2^{d-1}(d+1)\log n \\ &\quad = t\delta \\ &\qquad + ((L-1)(2C-(1+2^{-d}))(1-\delta)/2 \\ &\qquad\quad - (L-1/2)(1+2^{-d})) \\ &\qquad \times 2^d(d+1)\log n \end{aligned}$$

Finally, by substituting the chosen value $C = 1 + 3/(1-\delta)$ (and by using $L \geq 2$ and $d \geq 1$), we find that the second term in the last expression is nonnegative. Hence the relative Hamming distance between the words of γ and γ' is at least δ. □

In conclusion (using the option of constructing χ_2 by exhaustive search) we have the following theorem.

Theorem 7: The above "shingle" construction of χ can be constructed offline in sublinear time for any fixed dimension d, is online-efficient, and achieves any required relative distance $\delta < 1$. Its rate is proportional to $1/d^3$.

V. Trajectory Codes Have an Efficient Verification Procedure

In this section we show how to explicitly verify the distance property of any trajectory code using dynamic programming. This is in sharp contrast to tree codes, for which no such efficient verification procedure is known. Existence of an efficient verification procedure is important because our construction in the previous section has large constants. Using branch-and-bound methods along with the verification procedure might lead in practice to codes with better constants than are proven by our analysis.

Let $G = (V, E)$ be a graph with polynomial growth rate p. We show an algorithm that verifies that a finite time trajectory code $\chi : V \times \{1, 2, \ldots, T\} \to S$ has relative distance at least δ. The running time of the algorithm is polynomial in T.

The algorithm is a simple dynamic program. The dynamic programming table D is indexed by quintuples. Valid quintuples (x, y, z, t_0, t) are those for which $x, y, z \in V, t_0 + t \leq T$, and there exists a pair of twin trajectories (γ, γ') which begin at time t_0 at x, and such that at time t_0+t, γ ends at y while γ' ends at z. (In other words: $|\gamma| = |\gamma'| = t, \gamma(t_0) = \gamma'(t_0) = x, \gamma(t_0+t) = y, \gamma'(t_0+t) = z$, and for every $i > t_0, \gamma(i) \neq \gamma'(i)$.) We compute

$$D(x, y, z, t_0, t) = \min_{\text{twins}\gamma,\gamma'} h(\chi(\gamma), \chi(\gamma')).$$

Notice that the size of D can be loosely upper bounded by $(p(T))^3T^2$ which is polynomial in T. Clearly, upon completion of the computation of D, the relative distance of the code can be verified by checking if

$$D(x, y, z, t_0, t) \geq \delta t$$

for all valid quintuples (x, y, z, t_0, t).

The table D is computed by induction over t. For $t = 0$ the valid quintuples are $(x, x, x, t_0, 0)$ such that $t_0 \leq T$ and there is a length t_0 trajectory starting at x_0 and ending at x. For such valid quintuples we set $D(x, x, x, t_0, 0) = 0$. For $t > 0$, suppose we already computed all the valid entries of the form $(x, y, z, t_0, t-1)$. For every $t_0 \leq T - t$ and for every three

distinct nodes $x, y, z \in B(x_0, T)$ we compute the following. Let $\varepsilon \in \{0, 1\}$ be the indicator of $\chi(y, t_0 + t) \neq \chi(z, t_0 + t)$. Consider all pairs of nodes y', z' such that $(y', y), (z', z) \in E$ and $(x, y', z', t_0, t-1)$ is a valid quintuple. If no such pair exists, then (x, y, z, t_0, t) is not a valid quintuple. Otherwise, put

$$D(x, y, z, t_0, t) = \varepsilon + \min_{y', z'} \{D(x, y', z', t_0, t-1)\}.$$

This completes the description of the dynamic program.

Theorem 8: The dynamic program takes $\mathrm{poly}(T)$ time to execute, and it correctly computes $D(x, y, z, t_0, t)$ for all valid quintuples (x, y, z, t_0, t).

Proof: The number of quintuples (x, y, z, t_0, t) (valid or not) that are checked is at most $|B(x_0, T)|^3 T^2 \leq (p(T))^3 T^2$. The number of pairs y', z' that need to be examined in order to compute $D(x, y, z, t_0, t)$ is at most twice the maximum in-degree in the subgraph induced by $B(x_0, T)$. The proof of correctness is a trivial induction on t. □

REFERENCES

[1] N. Alon and J. H. Spencer, *The Probabilistic Method*, 2nd ed. New York: Wiley, 2000.

[2] N. Elia, "When Bode meets Shannon: Control-oriented feedback communication schemes," *IEEE Trans. Automat. Control*, vol. 49, no. 9, p. 1477, 2004.

[3] N. Elia and S. K. Mitter, "Stabilization of linear systems with limited information," *IEEE Trans. Automat. Control*, vol. 46, no. 9, pp. 1384–1400, Sep. 2001.

[4] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inf. Theory*, pp. 64–74, 1963.

[5] G. D. Forney, *Concatenated Codes*. Cambridge, MA: MIT Press, 1966.

[6] K. Goldberg, *Beyond Webcams: An Introduction to Online Robots*, R. S. , Ed. Cambridge, MA: MIT Press, 2002.

[7] J. Justesen, "A class of constructive, asymptotically good algebraic codes," *IEEE Trans. Inf. Theory*, vol. IT-18, pp. 652–656, Sep. 1972.

[8] D. Liberzon, "On stabilization of non-linear systems with limited information feedback," in *Proc. IEEE Conf. Dec. Control*, 2003, pp. 182–186.

[9] H. Marko, "The bidirectional communication theory—A generalisation of information theory," *IEEE Trans. Commun.*, vol. 21, no. 12, pp. 1345–1351, Dec. 1973.

[10] N. C. Martins, "Information Theoretic Aspects of the Control and Mode Estimation of Stochastic Systems," Ph.D. dissertation, Cambridge, MA, 2004, MIT.

[11] N. C. Martins and M. A. Dahleh, "Feedback control in the presence of noisy channels: "Bode-like" fundamental limitations of performance," *IEEE Trans. Automat. Control*, vol. 52, no. 7, pp. 1604–1615, Aug. 2008.

[12] J. Massey, "Causality, feedback and directed information," in *Proc. Int. Symp. Inf. Theory Its Applicat. (ISITA)*, 1990, pp. 303–305.

[13] A. Matveev and A. Savkin, "Comments on "control over noisy channels" and relevant negative results," *IEEE Trans. Automat. Control*, vol. 50, no. 12, pp. 2105–2110, Dec. 2005.

[14] S. K. Mitter, "Control with limited information: The role of systems theory and information theory," *Eur. J. Control*, vol. 7, pp. 122–131, Dec. 2000, (ISIT 2000 Plenary Talk, IEEE Information Theory Society Newsletter 50:1-23).

[15] S. K. Mitter, "System science: The convergence of communication, computation and control," in *Proc. 2002 Int. Conf. Control Applicat.*, 2002, vol. 1, pp. 18–20.

[16] R. M. Murray, *Control in an Information Rich World: Report of the Panel on Future Directions in Control, Dynamics and Systems*. New York: AFOSR, 2002.

[17] G. N. Nair and R. J. Evans, "State estimation via a capacity-limited communication channel," in *Proc. 36th IEEE Conf. Dec. Contr.*, 1997, pp. 866–871.

[18] G. N. Nair and R. J. Evans, "State estimation under bit-rate constraints," in *Proc. 37th IEEE Conf. Dec. Contr.*, 1998, pp. 251–256.

[19] M. Peczarski, "An improvement of the tree code construction," *Inf. Process. Lett.*, vol. 99, no. 3, pp. 92–95, 2006.

[20] B. Reiffen, Sequential Encoding and Decoding for the Discrete Memoryless Channel Res. Lab. of Electronics, M.I.T. Tech. Rep., 1960, Vol. 374.

[21] A. Sahai, "Anytime Information Theory," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Feb. 2001.

[22] L. J. Schulman, "Communication on noisy channels: A coding theorem for computation," in *Proc. 33rd Annu. Symp. Found. Comput. Sci.*, 1992, pp. 724–733.

[23] L. J. Schulman, "Deterministic coding for interactive communication," in *Proc. 25th Annu. Symp. Theory Comput.*, 1993, pp. 747–756.

[24] L. J. Schulman, "Coding for interactive communication," *IEEE Trans. Inf. Theory*, vol. 42, Special Issue on Codes and Complexity, no. 6, pp. 1745–1756, Nov. 1996.

[25] L. J. Schulman, Postscript to "Coding for Interactive Communication 2003 [Online]. Available: http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt

[26] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379, 623–423, 656, 1948.

[27] S. Tatikonda and S. Mitter, "Control over noisy channels," *IEEE Trans. Automat. Control*, vol. 49, no. 7, pp. 1196–1201, 2004.

[28] S. Tatikonda, A. Sahai, and S. Mitter, "Stochastic linear control over a communication channel," *IEEE Trans. Automat. Control*, vol. 49, no. 9, pp. 1549–1561, 2004.

[29] S. C. Tatikonda, "Control Under Communication Constraints," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Sep. 2000.

[30] H. S. Witsenhausen, "A counter-example in stochastic optimal control," *SIAM J. Control*, vol. 6, no. 1, pp. 131–147, 1968.

[31] W. S. Wong and R. W. Brockett, "Systems with finite communication bandwidth constraints-I: State estimation problems," *IEEE Trans. Automat. Control*, vol. 42, pp. 1294–1298, Sep. 1997.

[32] W. S. Wong and R. W. Brockett, "Systems with finite communication bandwidth constraints ii: Stabilization with limited information feedback," *IEEE Trans. Automat. Control*, vol. 44, pp. 1049–1053, May 1999.

[33] J. M. Wozencraft, Sequential Decoding for Reliable Communications Res. Lab. Electron., M.I.T. Tech. Rep., 1957, vol. 325.

[34] S. Yuksel and T. Basar, "Quantization and coding for decentralized LTI systems," in *Proc. IEEE Conf. Decision Control*, 2003, pp. 2847–2852.

Rafail Ostrovsky received the Ph.D. degree in computer science from the Massachusetts institute of Technology (MIT), Cambridge, in 1992, in the Theory of Computation Group.

He is a Professor of Computer Science and Mathematics at the University of California, Los Angeles (UCLA). He came to UCLA in 2003 from Bell Communications Research, where he was a Senior Research Scientist. Prior to beginning his career at Bell Communications Research, he was an NSF Mathematical Sciences Postdoctoral Research Fellow at UC Berkeley. His research centers on various issues in theoretical computer science, including cryptography, network algorithms, and high-dimensional search problems. He hold eight U.S. patents issued and over 130 papers published in refereed journals and conferences.

Dr. Ostrovsky is a member of the Editorial Board of *Algorithmica*; and the Editorial Board of the *Journal of Cryptology*; he serves on the Editorial and Advisory Board of the *International Journal of Information and Computer Security*. Professor Ostrovsky was invited as a Plenary Speaker at a conference organized by FBI in 2009, and was invited as a Keynote Speaker for Public Key Cryptography International Conference in 2007. In addition to numerous invitations to special issues dedicated to top-rated STOC/FOCS articles, his awards include the Best Paper Award of the 2008 International Conference on Computing and Combinatorics (COCOON-2008); 2006 and 2005 Xerox Corporate Innovation Faculty Awards; 2006 IBM Faculty Award; 2006 Xerox Corporation Distinguished Lecture Series; 2005 Distinguished Cryptographer of the Year Lecture Series NTT Labs, Japan; OKAWA Foundation 2004 Research Award; three SAIC Awards for the best published work of the year (1999, 2001, 2002) in computer science and mathematics; the 1996 Bellcore Prize for excellence in research; and 1993 Henry Taub Prize. At UCLA, he heads security and cryptography multi-disciplinary Research Center (http://www.cs.ucla.edu/security/) at Henry Samueli School of Engineering and Applied Science.

Yuval Rabani studied in Tel Aviv University, Israel, and received the Ph.D. degree in computer science in 1992.

Since 1995, he has been on the faculty of the Technion—Israel Institute of Technology, Haifa. He has also held visiting appointments at Cornell University, University of California, Los Angeles, and Caltech. His research interests include: computational aspects of metric geometry, combinatorial approximation algorithms, network optimization, and online computing.

Leonard J. Schulman received the B.Sc. degree in mathematics in 1988 and the Ph.D. degree in applied mathematics in 1992, both from the Massachusetts Institute of Technology, Cambridge.

Since 2000, he has been on the faculty of the California Institute of Technology, Pasadena. He has also held appointments at UC Berkeley, the Weizmann Institute of Science, the Georgia Institute of Technology, and the Mathematical Sciences Research Institute. His research is in several overlapping areas: algorithms and communication protocols; combinatorics and probability; coding and information theory; quantum computation.

Dr. Schulman received the MIT Bucsela prize in mathematics, an NSF mathematical sciences postdoctoral fellowship, an NSF CAREER award, and the IEEE Schelkunoff prize. He is the director of the Caltech Center for the Mathematics of Information and is on the faculty of the Institute for Quantum Information.