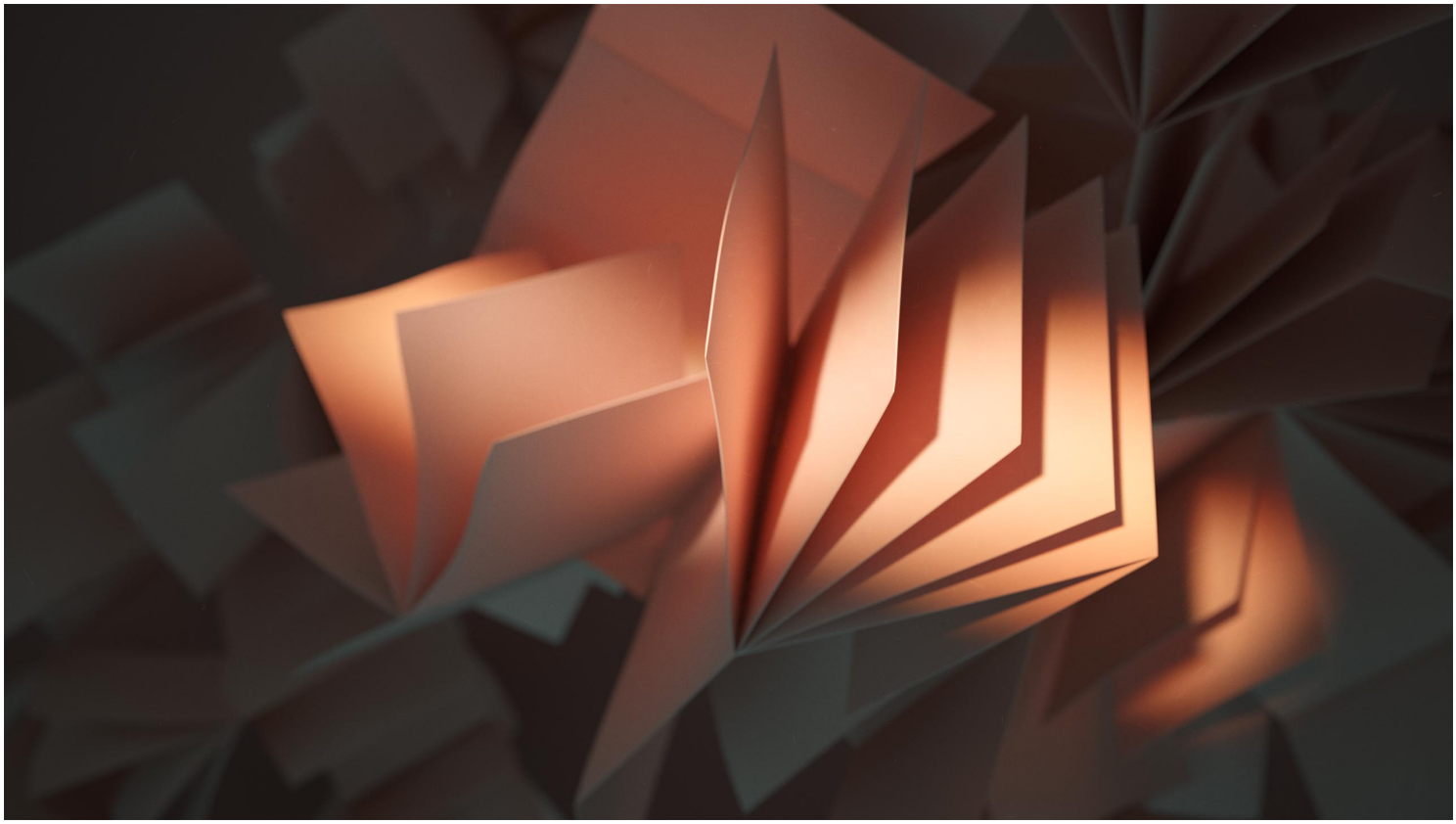
INFORMATION THEORY

Researchers Defeat Randomness to Create Ideal Code

By MORDECHAI RORVIG

November 24, 2021

By carefully constructing a multidimensional and well-connected graph, a team of researchers has finally created a long-sought locally testable code that can immediately betray whether it's been corrupted.



To create an optimal method for encoding information, researchers represented it in a graph that takes the form of a richly interconnected web of booklets that explodes outward. Each square in the graph represents one single bit of information from a message.

—

Olena Shmahalo for Quanta Magazine

Suppose you are trying to transmit a message. Convert each character into bits, and each bit into a signal. Then send it, over copper or fiber or air. Try as you might to be as careful as possible, what is received on the other side will not be the same as what you began with. Noise never fails to corrupt.

In the 1940s, computer scientists first confronted the unavoidable problem of noise. Five decades later, they came up with an elegant approach to sidestepping it: What if you could encode a message so that it would be obvious if it had been garbled before your recipient even read it? A book can't be judged by its cover, but this message could.

They called this property local testability, because such a message can be tested super-fast in just a few spots to ascertain its correctness. Over the next 30 years, researchers made substantial progress toward creating such a test, but their efforts always fell short. Many thought local testability would never be achieved in its ideal form.

Now, in a preprint released on November 8, the computer scientist Irit Dinur of the Weizmann Institute of Science and four mathematicians, Shai Evra, Ron Livne, Alex Lubotzky and Shahar Mozes, all at the Hebrew University of Jerusalem, have found it.

"It's one of the most remarkable phenomena that I know of in mathematics or computer science," said Tom Gur of the University of Warwick. "It's been the holy grail of an entire field."

Their new technique transforms a message into a super-canary, an object that testifies to its health better than any other message yet known. Any corruption of significance that is buried anywhere in its superstructure becomes apparent from simple tests at a few spots.

Irit Dinur of the Weizmann Institute of Science helped construct an error-correcting code with a combination of ideal properties, which remain constant even as codewords scale up in size.

—

Hadar Dveer Benyamini

"This is not something that seems plausible," said Madhu Sudan of Harvard University. "This result suddenly says you can do it."

Most prior methods for encoding data relied on randomness in some form. But for local testability, randomness could not help. Instead, the researchers had to devise a highly nonrandom graph structure entirely new to mathematics, which they based their new method on. It is both a theoretical curiosity and a practical advance in making information as resilient as possible.

Coding 101

Noise is ubiquitous in communication. To analyze it systematically, researchers first represent information as a sequence of bits, 1s and 0s. We can then think of noise as a random influence that flips certain bits.

There are many methods for dealing with this noise. Consider a piece of information, a message as short and simple as 01. Modify it by repeating each piece of it — each bit — three times, to get 000111. Then, even if noise happens to corrupt, say, the second and sixth bits — changing the message to 010110 — a receiver can still correct the errors by taking majority votes, twice (once for the 0s, once for the 1s).

Such a method of modifying a message is called a code. In this case, since the code also comes with a procedure for fixing errors, it is called an error-correcting code. Codes are like dictionaries, each one defining a certain set of codewords, such as 000111.

To work well, a code must have several properties. First, the codewords in it should not be too similar: If a code contained the codewords 0000 and 0001, it would only take one bit-flip's worth of noise to confuse the two words. Second, codewords should not be too long. Repeating bits may make a message more durable, but they also make it take longer to send.

These two properties are called distance and rate. A good code should have both a large distance (between distinct codewords) and a high rate (of transmitting real information). But how can you obtain both properties at once? In 1948, Claude Shannon showed that any code whose codewords were simply chosen at random would have a nearly optimal trade-off between the two properties. However, choosing codewords completely at random would make for an unpredictable dictionary that was excessively hard to sort through. In other words, Shannon showed that good codes exist, but his method for making them didn't work well.

"It was an existential result," said Henry Yuen of Columbia University.

Over the next 40 years, computer scientists worked to figure out nonrandom recipes for arranging bits that approached the random ideal. By the late 1980s, their codes were used in everything from CDs to satellite transmissions.

In 1990, researchers formulated the idea of local testability. But this property was different. If you picked a code at random, as Shannon advised, there was no way it could be a locally testable code. These were the albino butterflies in the universe of random codes — beautiful, if they existed.

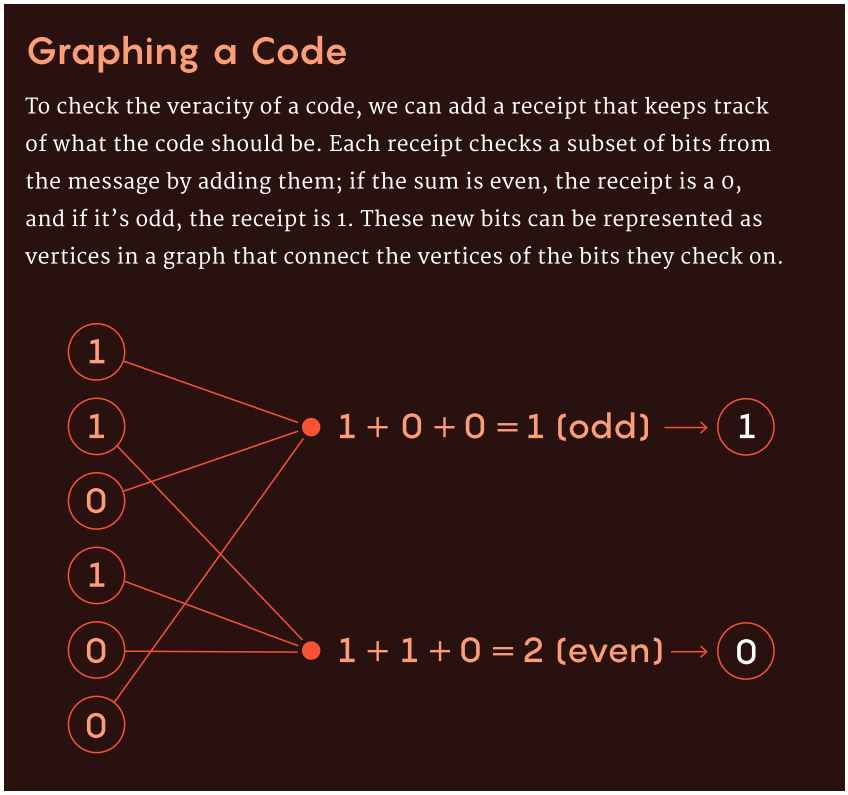
"You actually have to work much harder to even show that they exist," said Yuen. "Never mind coming up with an explicit example."

Graphs as Codes

To understand why testability is so hard to obtain, we need to think of a message not just as a string of bits, but as a mathematical graph: a collection of vertices (dots) connected by edges (lines). This equivalence has been central to the understanding of codes ever since the first clever codes were created by Richard Hamming, two years after Shannon's result. (The graphical perspective became particularly influential after a 1981 paper by R. Michael Tanner.)

Hamming's work set the stage for the ubiquitous error-correcting codes of the 1980s. He came up with a rule that each message should be paired with a set of receipts, which keep an account of its bits. More specifically, each receipt is the sum of a carefully chosen subset of bits from the message. When this sum has an even value, the receipt is marked 0, and when it has an odd value, the receipt is marked 1. Each receipt is represented by one single bit, in other words, which researchers call a parity check or parity bit.

Hamming specified a procedure for appending the receipts to a message. A recipient could then detect errors by attempting to reproduce the receipts, calculating the sums for themselves. These Hamming codes work remarkably well, and they are the starting point for seeing codes as graphs and graphs as codes.



—

Samuel Velasco/Quanta Magazine

"For us, to think about a graph and to think about a code is the same thing," said Dana Moshkovitz of the University of Texas, Austin.

To make a graph from a code, start with a codeword. For each bit of information, draw a vertex (or node), called a digit node. Then draw a node for each of the parity bits; these are called parity nodes. Finally, draw lines from each parity node to the digit nodes that are supposed to add up to form its parity value. You have just created a graph from a code.

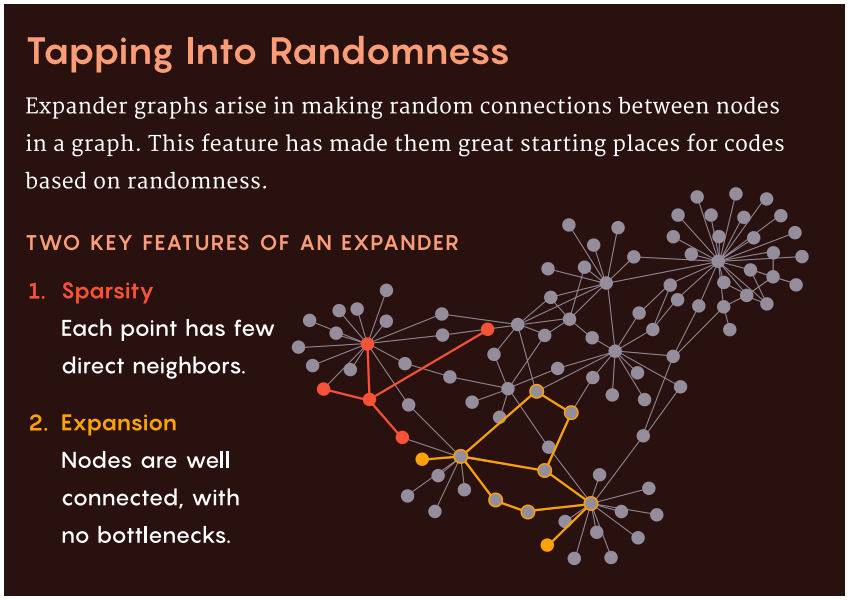
Seeing codes and graphs as equivalent became integral to the art of making codes. In 1996, Michael Sipser and Daniel Spielman used the method to create a breakthrough code out of a kind of graph called an expander graph. Their code still couldn't provide local testability, but it proved optimal in other ways and eventually served as the basis for the new results.

Expanding the Possibilities

Expander graphs are distinguished by two properties that can seem contradictory. First, they are sparse: Each node is connected to relatively few other nodes. Second, they have a property called expandedness — the reason for their name — which means that no set of nodes can be bottlenecks that few edges pass through. Each node is well connected to other nodes, in other words — despite the scarcity of the connections it has.

"Why should such an object ever exist?" said Evra. "It's not so far-fetched to think that if you're sparse, then you're not so connected."

But expander graphs are actually surprisingly easy to create. If you construct a graph in a random way, choosing connections at random between nodes, an expander graph will inevitably result. They're like a source of pure, unrefined randomness, making them natural building blocks for the good codes that Shannon pointed toward.



Samuel Velasco/Quanta Magazine

Sipser and Spielman worked out how to turn an expander graph into a code. The codewords they came up with were built from many much-shorter words produced by a Hamming code, which they called a small code. The bits of their codewords were represented as the edges of the expander graph. And all the receipts for the small code were represented at each node.

In effect, Sipser and Spielman showed that if you define the small codes at each node with good properties, then because the graph is so well connected, those properties propagate to the global code. This propagation gave them a way to create a good code.

"Expansion, expansion and again expansion," said Evra. "That's the secret for success."

However, local testability was not possible. Suppose that you had a valid codeword from an expander code, and you removed one receipt, or parity bit, from one single node. That would constitute a new code, which would have many more valid codewords than the first code, since there would be one less receipt they needed to satisfy. For someone working off the original code, those new codewords would satisfy the receipts at most nodes — all of them, except the one where the receipt was erased. And yet, because both codes have a large distance, the new codeword that seems correct would be extremely far from the original set of codewords. Local testability was simply incompatible with expander codes.

To obtain testability, researchers would have to figure out how to work against the randomness that had formerly been so helpful. In the end, the researchers went where randomness could not: into higher dimensions.

The Opposite of Random

It wasn't always clear they could make it. Local testability was achieved by 2007, but only at the cost of other parameters, like rate and distance. In particular, these parameters would degrade as a codeword became large. In a world constantly seeking to send and store larger messages, these diminishing returns were a major flaw. (Though in practice, even the existing locally testable codes were already more powerful than most engineers needed to use.)

In 1996, Michael Sipser (left) and Daniel Spielman created a code based on expander graphs that had an excellent combination of properties, but it failed to be at all locally testable.

—

Bryce Vickmark; John D. and Catherine T. MacArthur Foundation

The hypothesis that a code could be found with optimal rate, distance and local testability — which all stayed constant even as messages were scaled up — came to be known as the c^3 conjecture. The prior results made some researchers think that a solution was inevitable. But progress started to slow, and other results suggested the conjecture might be false.

"Many in the community thought that it was a dream that was probably too good to be true," said Gur. "Things looked quite grim."

But in 2017, a new source of ideas emerged. Dinur and Lubotzky began working together while attending a yearlong research program at the Israel Institute for Advanced Studies. They came to believe that a 1973 result by the mathematician Howard Garland might hold just what computer scientists sought. Whereas ordinary expander graphs are essentially one-dimensional structures, with each edge extending in only one direction, Garland had created a mathematical object that could be interpreted as an expander graph that spanned higher dimensions, with, for example, the graph's edges redefined as squares or cubes.

Garland's high-dimensional expander graphs had properties that seemed ideal for local testability. They must be deliberately constructed from scratch, making them a natural antithesis of randomness. And their nodes are so interconnected that their local characteristics become virtually indistinguishable from how they look globally.

"To me, high-dimensional expander graphs are a wonder," said Gur. "You make a tiny tweak to one part of the object and everything changes."

Lubotzky and Dinur began trying to create a code based on Garland's work that might solve the c^3 conjecture. Evra, Livne and Mozes soon joined the team, each of them experts in different aspects of high-dimensional expanders.

Soon they were presenting their work in seminars and talks, but not everyone was convinced that the theory of high-dimensional expanders would pave the way forward. To understand it at all required ascending a steep learning curve.

"At the time it seemed like space-age technology, sophisticated and exotic mathematical tools never seen before in computer science," said Gur. "It seemed like overkill."

In 2020, the researchers got stuck, until they realized that they could get by without relying on the most complicated new tools. The inspiration they had gained from high-dimensional expanders was enough.

Propagating Mistakes

In their new work, the authors figured out how to assemble expander graphs to create a new graph that leads to the optimal form of locally testable code. They call their graph a left-right Cayley complex.

As in Garland's work, the building blocks of their graph are no longer one-dimensional edges, but two-dimensional squares. Each information bit from a codeword is assigned to a square, and parity bits (or receipts) are assigned to edges and corners (which are nodes). Each node therefore defines the values of bits (or squares) that can be connected to it.

To get a sense of what their graph looks like, imagine observing it from the inside, standing on a single edge. They construct their graph such that every edge has a fixed number of squares attached. Therefore, from your vantage point you'd feel as if you were looking out from the spine of a booklet. However, from the other three sides of the booklet's pages, you'd see the spines of new booklets branching from them as well. Booklets would keep branching out from each edge ad infinitum.

"It's impossible to visualize. That's the whole point," said Lubotzky. "That's why it is so sophisticated."

Crucially, the complicated graph also shares the properties of an expander graph, like sparseness and connectedness, but with a much richer local structure. For example, an observer sitting at one vertex of a high-dimensional expander could use this structure to straightforwardly infer that the entire graph is strongly connected.

“What’s the opposite of randomness? It’s structure,” said Evra. “The key to local testability is structure.”

To see how this graph leads to a locally testable code, consider that in an expander code, if a bit (which is an edge) is in error, that error can only be detected by checking the receipts at its immediately neighboring nodes. But in a left-right Cayley complex, if a bit (a square) is in error, that error is visible from multiple different nodes, including some that are not even connected to each other by an edge.

In this way, a test at one node can reveal information about errors from far away nodes. By making use of higher dimensions, the graph is ultimately connected in ways that go beyond what we typically even think of as connections.

In addition to testability, the new code maintains rate, distance and other desired properties, even as codewords scale, proving the c^3 conjecture true. It establishes a new state of the art for error-correcting codes, and it also marks the first substantial payoff from bringing the mathematics of high-dimensional expanders to bear on codes.

“It’s a completely new way of looking at these objects,” said Dinur.

Practical and theoretical applications should soon follow. Different forms of locally testable codes are now being used in decentralized finance, and an optimal version will allow even better decentralized tools. Furthermore, there are totally different theoretical constructs in computer science, called probabilistically checkable proofs, which have certain similarities with locally testable codes. Now that we’ve found the optimal form of the latter, record-breaking versions of the former seem likely to appear.

Ultimately, the new code marks a conceptual milestone, the greatest step yet beyond the boundaries for codes set by randomness. The only question left is whether there are any true limits to how well information can be forged.