Supplementary Information for

HEDGES Error-Correcting Code for DNA Storage Corrects Indels and Allows Sequence Constraints

William H. Press, John A. Hawkins, Stephen Knox Jones Jr, Jeffrey M. Schaub, and Ilya J. Finkelstein

William H. Press.
E-mail: wpress@cs.utexas.edu

This PDF file includes:

Supporting Information Text

Supplementary Text

A. Example System Design with Outer Code. For cost and efficiency, both DNA synthesis and DNA sequencing employ massive parallelism. That is, many short sequences, each of length hundreds to thousands of bases, are written (synthesized) or read (sequenced) simultaneously. While the length of a single synthesis or read will increase as technology improves, it is unlikely that the great advantage of parallelism will ever be superceded. This being the case, the basic units of our design are individual strands of length 10^2–10^4.

To connect with our use of a Reed-Solomon outer code, here RS(255,223), we define a "DNA packet" as an ordered set of 255 DNA strands. When any one strand in the set is decoded with HEDGES, it produces a message fragment of length L bytes (say), now having high probability of being perfectly synchronized. Each 255 correctly ordered message fragments form a "message packet", as illustrated in main text Figure 1C. There can be any number of message packets in a total communication.

The Reed-Solomon code is applied across the strands (interleaved). This enables it to protect against missing strands—"erasures" to coding theorists—as well as correcting any residual substitution errors that were not corrected by HEDGES. Different from previous investigations, we apply the RS code diagonally across the strands (see Figure). This increases the resistance to any failure of synthesis or sequencing to produce full-length strands. It also ameliorates the effect of the observed tendency for error rates to be higher at the ends of strands.

It is an important point that the Reed-Solomon code can only be applied after the strands in a packet are identified as being from one particular packet (out of an assumed pool of many packets, perhaps millions) and are correctly ordered. This implies that a packet's identification number and a strand's serial number within the packet (both shown as shaded green in the Figure) cannot themselves be RS protected. We therefore protect them by the different technique ("salt protection") that was described in Methods. Salt protection has the effect of turning uncorrectable errors in the identification/serial bytes into erasures in the message bytes—which are correctable by RS.

Summarizing, the main points that relate to the use of HEDGES as an inner code are these: (1) We don't need to decode strands of arbitrary length, but only of some known uncorrupted length L. (2) Recovering synchronization has the highest priority. (3) Known erasures are less harmful than unknown substitutions, because RS can correct twice as many erasures as substitution errors. (4) Burst errors within a single byte are less harmful than distributed bit errors, because RS corrects a byte at a time. (5) Within the RS code's capacity for byte errors and erasures, residual errors will be fully corrected by the outer code, yielding an error-free message.

B. Details of the encoding algorithm. Elaborating on the description in the main text, let S_i denote an arbitrary s-bit value ("salt") that can depend on i but is known to both sender and receiver,

$$S_i = \text{known} \in \mathbb{Z}_2^{\otimes s}$$

(The purpose of the salt is discussed in the main text.) Denote the low-order q bits of the bit position index i by

$$I_i \equiv i \pmod{2^q}$$

Let B_i denote the p previous concatenated bits

$$B_i \equiv [b_{i-p} b_{i-p+1} \cdots b_{i-1}] \in \mathbb{Z}_2^{\otimes p}$$

Finally, let $F(S, I, B)$ be a deterministic hash function from $p+q+s$ bits to 2 bits

$$F(S, I, B): \quad \mathbb{Z}_2^{\otimes (p+q+s)} \to \mathbb{Z}_4$$

Then the formula for encoding is

$$C_i = K_i + b_i = F(S_i, I_i, B_i) + b_i \pmod 4 \qquad [1]$$

Main text Figure 1D shows the algorithm graphically.

Typical values that we use are $p = 8$, $q = 10$, $s = 46$, so that $p + q + s = 64$ bits, a convenient value for input to the hash. For the hash function we use the low order 2 bits from the *Numerical Recipes* (1) function `Ranhash.int64()`, because it is very fast and will occur in the inner loop of the decode algorithm.

C. Details of the decoding algorithm. We can formalize the discussion in the main text as follows. Let $H{:=}[i, b_i, B_i, k]$ denote the joint hypothesis that the values i, b_i, B_i are all correct and synchronize to the observed codestream character C'_k through equation Eq. (1). As a node in the search tree, the hypothesis $H{:=}[i, b_i, B_i, k]$ spawns six child hypotheses, each of which can be scored with additional penalty ΔP (to be added to their common parent's accumulated penalty) as follows:

$$\begin{aligned}
H{:=}[i+1, \{0,1\}, B_{i+1}, k]: \\
&\Delta P = P_{\text{del}} \\
H{:=}[i+1, \{0,1\}, B_{i+1}, k+1]: \\
&\Delta P = (P_{\text{ok}} \text{ if } C = C' \text{ else } P_{\text{sub}}) \\
H{:=}[i+1, \{0,1\}, B_{i+1}, k+2]: \\
&\Delta P = (P_{\text{ins}} + P_{\text{ok}} \text{ if } C = C' \text{ else } P_{\text{ins}} + P_{\text{sub}})
\end{aligned} \qquad [2]$$

Here $P_{\rm sub}, P_{\rm ins}, P_{\rm del}$ can be thought of as respectively the log probability penalties for substitution, insertion, or deletion errors (but see below for additional nuance). $P_{\rm ok}$ is the penalty or, if negative, reward, for an agreement between the computed and received codestream characters C and C'. In the comparison notated above as $C = C'$, the index of C is the first parameter in the hypothesis H, while the index of C' is the last parameter in H. Note that a child node's B_{i+1} is always computable from its parent's B_i and b_i.

How can we practically search this huge tree? A conceptual starting point is the famous A* search algorithm (2), a best-first (that is, "greedy") search utilizing a heap data structure. A* assigns a heuristic cost to every node that is the sum of its actual cost plus a quantity less than or equal to the smallest possible additional cost that it can incur in reaching the goal. (For a tree of constant depth, this is equivalent to adding a reward for every step taken closer to the leaf nodes, i.e., a negative constant $P_{\rm ok}$ above.) Main text Figure 1F shows the logical flow of an A* search, and also the HEDGES decode algorithm.

Provably, A* always finds the best path. For our application, unfortunately, this guaranteed result is exponentially slow, because actual errors along the true path cause too many spawned hypotheses to be revisited; and because its termination criterion is too restrictive, again leading to too many spawned hypotheses.

To ameliorate these problems we make two heuristic modifications of A*: First, we allow $P_{\rm ok}$ to be more negative than that sanctioned by A* and tune its value heuristically. While we thus lose the guarantee of finding exactly the shortest path, we heuristically encourage the search not to revisit earlier hypotheses after a sufficiently lengthy run of successes along one particular chain. Second, we adopt a "first past the post" termination criterion. That is, the first chain of hypotheses to decode the required L bytes of message wins. It is not obvious (or, by us, provable) that these heuristics should result in a workable or efficient algorithm, but for the success of our numerical experiments.

D. Choice of, And Trade-Offs Among, Parameters. It might at first seem that bigger is better for both q and p, but this is not the case. Restricting p to a smaller value better allows the heap search to recover from previous errors, basically by finding a clever acasual (i.e., "wrong") path that coincidentally puts it back on track. As for q, restricting it to a smaller value could be useful in case one desires the capability of jumping into the middle of an undecoded message: The heap can then be initialized with all possible values of I and B (cf. main text Figure 1D). For our concatenated design (Supplementary Information) this is not a necessary, or useful, capability, however. For the validation experiments described above in Results, we took $q = 10$, $r = 8$, $n = 16$ or 24.

For decoding, we need to know the encoding parameters, and must now also choose values for $P_{\rm sub}, P_{\rm del}, P_{\rm ins}$, and $P_{\rm ok}$. While, conceptually, these are negative log probabilities of the occurrence of the different kinds of errors (which can be known only after the fact), we adopt a more empirical approach. First, we take $P_{\rm sub} = P_{\rm del} = P_{\rm ins}$ to give the HEDGES decoding algorithm equal robustness against all three kinds of errors. Second, we note that the search for shortest path is invariant under applying the same linear (or affine) transformation to all four P's. So, without loss of generality, we may take $P_{\rm sub} = P_{\rm del} = P_{\rm ins} = 1$, leaving $P_{\rm ok}$ as the only free parameter. We determine optimal (or at least good) values for $P_{\rm ok}$ by numerical experiment. We find that the optimal $P_{\rm ok}$ depends only negligibly on the encoding parameters q and p, and only slightly on the length L of the strand, but it does depend on the code rate. Good values for various code rates are given in the third column of Table **3**.

Implicitly, the choice of $P_{\rm ok}$ reflects a tradeoff between computational workload and decode failure probability. $P_{\rm ok}$ that is too negative results in too greedy a search, which is fast but can get stuck in a blind alley that requires us to declare the rest of the strand as an erasure (hence its dependence on strand length). On the other hand, $P_{\rm ok}$ that is insufficiently negative results in a too large, potentially exponential, expansion of the size of the heap. Happily, there is an accessible range of workable values. Changes of ∼10% in $P_{\rm ok}$ matter little, and our values are implicitly tuned for best performance on strand lengths in the range ∼100 to ∼1000.

Above, we limited the guesses for skew Δ to only $\{-1, 0, 1\}$ so as to limit the expansion of the heap. (Even allowing skew $\Delta = \{-2, -1, 0, 1, 2\}$ explodes the heap size and computation time). Runs of deletions can correctly be decoded with consecutive deletion hypotheses. However, this skew limit results in more than one consecutive insertion being improperly scored. For example, without the possibility of skew $\Delta = +2$, the shortest available path through two insertions $\ldots II \ldots$ incurs a spurious substitution $\ldots ISI \ldots$, with probability $\geq \frac{1}{4}$ of the randomly inserted characters still allowing a correct decode path (e.g. if the second inserted base matches the following character). In practice, this makes little difference, because double insertions are significantly less common than single insertions, and because other, completely incorrect, paths score much worse.

Adaptive choice of $H_{\rm limit}$. As commented in the main text, the decode heap size $H_{\rm limit}$ is not an irrevocable choice. Strands that fail to decode can be retried with a larger value of $H_{\rm limit}$. Figure S6 shows failure rates for different values of $H_{\rm limit}$. One sees that in the typical case, increasing $H_{\rm limit}$ by a factor eight gives roughly one order of magnitude decrease in decode failure probability.

Entropy loss from constraints

As noted in the main text, imposing sequence constraints on allowed output sequences does not change the code rate, r, of HEDGES encoding. Rather, the error-correcting capacity is slightly reduced, corresponding to the overall entropy loss resulting from the selected constraints. In order to characterize this effect, we here calculate the entropy loss resulting from the two most common types of sequence constraints, homopolymer runs and GC content.

E. Homopolymer run constraints. We here find the entropy loss due to restricting homopolymer runs to a maximum length R. Let n be a given sequence length and let S_n be the number of length-n sequences satisfying a given constraint. The entropy per character of length-n sequences is given by

$$H_n = \frac{1}{n}\,\log_2(S_n) \tag{3}$$

Let $v_{n,i}$ be the number of sequences of length n such that the final i bases are the same, and the base at position $n-i-1$ is different from the final i bases. We say such a sequence has end run length i. For example, the sequece $GGGAA$ is length 5 with end run length 2, so corresponds to $v_{5,2}$. Finally, write the vector of $v_{n,i}$ over i as $\boldsymbol{v}_n$.

A simple system of linear equations relates $\boldsymbol{v}_n$ to $\boldsymbol{v}_{n-1}$. For every sequence in $v_{n-1,i}$, one of the four possible next bases will extend the end run (add to $v_{n,i+1}$) and the other three possible next bases will break the run (add to $v_{n,1}$). That is,

$$\begin{aligned}\boldsymbol{v}_n &= \begin{pmatrix} 3 & 3 & 3 & \cdots & 3 & 3 \\ 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & 0 \end{pmatrix} \begin{pmatrix} v_{n-1,1} \\ v_{n-1,2} \\ v_{n-1,3} \\ v_{n-1,4} \\ \vdots \\ v_{n-1,R} \end{pmatrix} \\ &= M\boldsymbol{v}_{n-1} = M^{n-1}\boldsymbol{v}_1 \end{aligned} \tag{4}$$

where $\boldsymbol{v}_1 = (\,4,\ 0,\ 0,\ \cdots,\ 0\,)^T$ and $S_n = \sum_{i=1}^{R} v_{n,i}$. The filtering of sequences with long runs happens by their "falling off the bottom" of the matrix equation, as it were.

Lemma: H_n converges to $\log_2(\lambda^*)$, where λ^* is the largest eigenvalue of M

Proof: This is true by the same reasoning as the power method of finding eigenvalues: for the largest eigenvalue λ^* and corresponding eigenvector u^* of M,

$$M\left(\frac{M^n x}{\|M^n x\|}\right) \to \lambda^*\left(\frac{M^n x}{\|M^n x\|}\right) \quad \forall\ x \not\perp u^* \tag{5}$$

Even a single application of M always makes $v_{n,1}$ the largest value of $\boldsymbol{v}_n$, so $u_1^* \neq 0$ and thus $\boldsymbol{v}_1 \not\perp u^*$. The average of a converging sequence converges to the same value, so it is sufficient to find the limit of H_n^{last}, the entropy of the last character. Hence,

$$\begin{aligned} H_n^{last} &= \log_2(S_n) - \log_2(S_{n-1}) \\ &= \log_2\left(\frac{\sum_i (M\boldsymbol{v}_{n-1})_i}{\sum_i v_{n-1,i}}\right) \\ &\to \log_2(\lambda^*) \end{aligned}$$

Recognizing the matrix M as the transition matrix of the linear recurrence relationship,

$$a_n = 3\sum_{i=0}^{n-1} a_{n-1-i} \tag{6}$$

we can read off the characteristic polynomial by inspection. Hence, for maximum homopolymer run length R,

$$\begin{aligned} H(R) &= \lim_{n\to\infty} H_n(R) \\ &= \log_2\left(\text{LargestRoot}\left[\lambda^R - 3\sum_{i=0}^{R-1}\lambda^i = 0\right]\right) \end{aligned} \tag{7}$$

The largest root here is the effective number of possible characters per position, which tends toward 4 as R gets large. Letting $\lambda^* = 4+\varepsilon$, one finds that as R gets large, $\varepsilon \approx -\frac{3}{4^R}$. And hence, the fractional entropy loss from 2 bits of entropy per character, $L(R)$, is

$$L(R) \equiv \frac{2-H(R)}{2} \approx \frac{3}{8\ln 2}4^{-R} \tag{8}$$

Figure S4A shows several numerically solved values of $L(R)$, as well as this asymptotic approximation.

F. Arbitrary constraints on subsequences. The results from the previous section can be generalized in a straightforward manner to arbitrary constraints on (short) subsequences as follows.

First, generate the list of all allowed subsequences of length w, the maximum constraint width. Consider each allowed sequence to be a node in the graph G of sequence paths and create a directed edge from one allowed subsequence to any allowed subsequence which could follow it by sliding the constraint window one base to the right. I.e., the allowed sequence $s_1 s_2 \dots s_w$ has a directed edge to each $s_2 \dots s_w N$ that is still an allowed sequence. By construction, there is a bijection between allowed full length sequences of any length greater than w and paths through this graph.

Let M be the adjacency matrix of G and let $\boldsymbol{v}_w = (1, 1, 1, \dots, 1)^T$ initialize the fact that each allowed sequence ends exactly one sequence of length w, itself. Then as above $\boldsymbol{v}_{n+1} = M\boldsymbol{v}_n$ and $H_n \to \log_2(\lambda^*)$. We do not in general have a nice characteristic polynomial as before, but the largest eigenvalue can be found numerically. The principal limit of this method is that the number of allowed subsequences must be relatively small for computationally tractability.

G. GC content constraints. For GC content contstraints, we proceed as described in the previous section. But we can exploit the fact that under this constraint $A = T$ and $C = G$. We can thus consider only sequences with A and C and then add one bit of entropy per character for the entropy of A/T or C/G. This reduces the number of allowed sequences, and hence the side length of M, by a factor of 2^w. Figure S4B shows the fractional entropy loss for all GC content constraints in windows up to width $w = 20$.

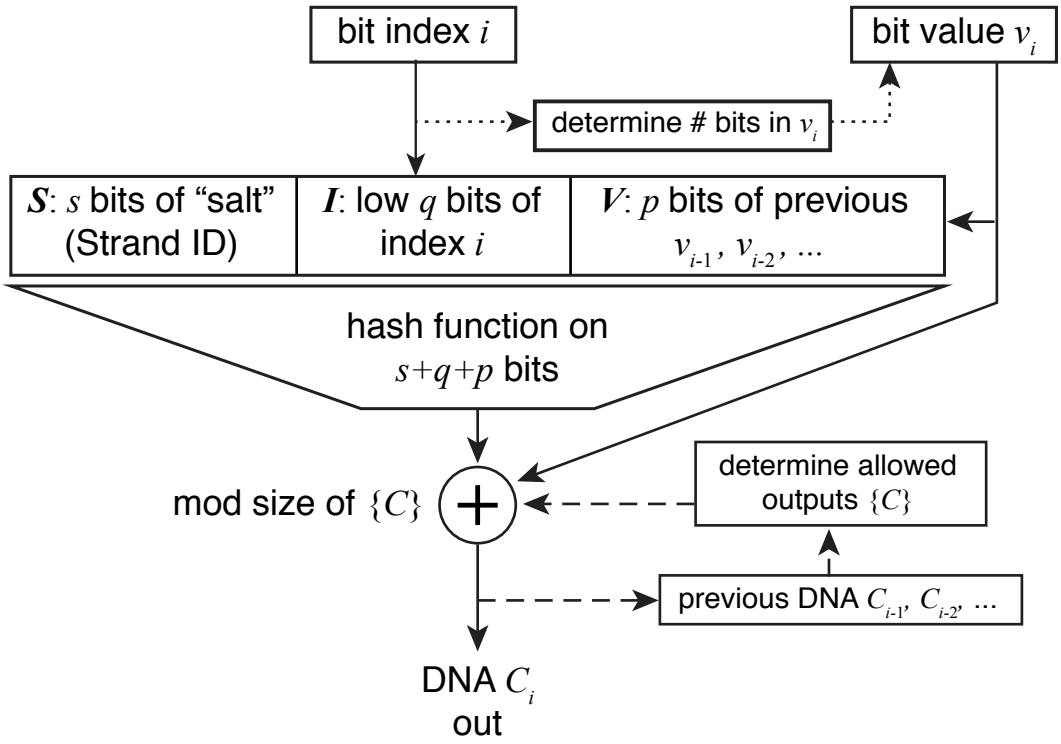

Fig. S1. Full HEDGES algorithm, including steps for variable code rates (dotted lines) and sequence constraints (dashed lines).

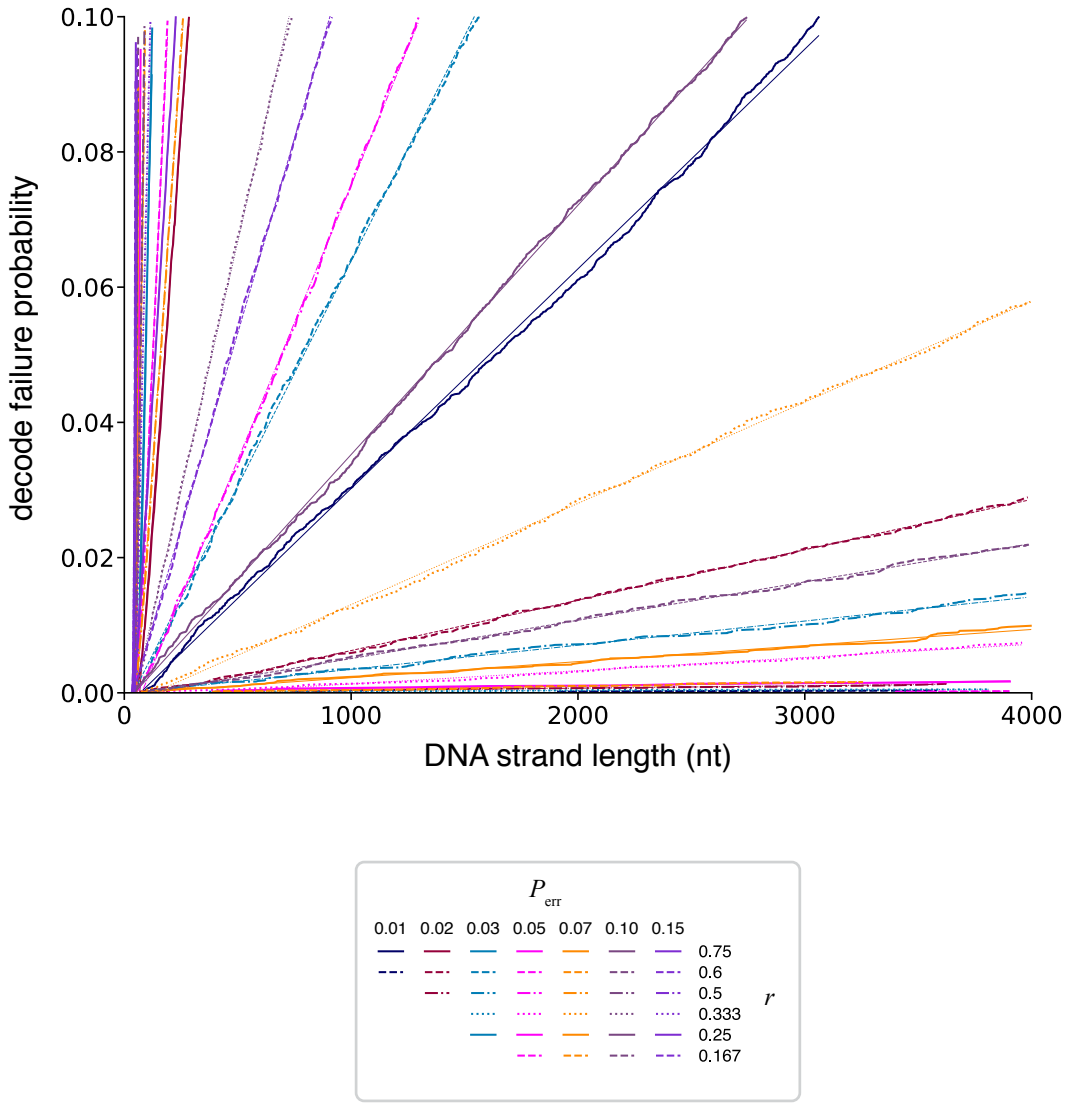

Fig. S2. Cumulative decode failure probability as a function of DNA strand length for the simulations described in the text. A separate simulation is performed for each combination of DNA error rate P_{err} and code rate r, as shown in the legend. The simulation results are well fit by straight lines (narrow lines as shown), indicating that the probability of decode failure is approximately a constant (the slope) per nucleotide position along each strand. The small offset ($\sim$50 to $\sim$100 nt) from the origin is an artifact of the maximum allowed heap size: A decode does not signal failure until its heap is exhausted.

Total DNA error rate (with sub = ins = del):

code rate:	0.01	0.02	0.03	0.05	0.07	0.10	0.15
	$<10^{-6}$	$<10^{-6}$	$<10^{-6}$	$<10^{-6}$	0.000001	0.000004	0.000019
	$<10^{-6}$	0.000002	0.000002	0.000004	0.000006	0.000019	0.000096
0.166	$>10^{7}$	$>10^{7}$	$>10^{7}$	$>10^{7}$	1800000	180000	8600
	0.0021	0.0024	0.0023	0.0014	0.0122	0.1123	2.2548
	1.3E+60	8.6E+58	1.8E+59	1.3E+63	9.4E+46	4.5E+30	2.42E+09
	$<10^{-6}$	0.000004	0.000006	0.000014	0.000036	0.000140	0.000910
	0.000004	0.000017	0.000024	0.000058	0.000130	0.000500	0.003000
0.250	$>10^{7}$	$>10^{7}$	7100000	2100000	280000	25000	1700
	0.0029	0.0062	0.0088	0.0238	0.1015	0.8867	11.9555
	3.3E+57	8.7E+51	2.6E+49	1.2E+42	2.5E+31	5.2E+15	7.32E+00
	0.000014	0.000030	0.000053	0.000170	0.000500	0.002200	0.013000
	0.000047	0.000100	0.000180	0.000540	0.001400	0.005800	0.032000
0.333	$>10^{7}$	6300000	3100000	280000	49000	6200	630
	0.014	0.029	0.052	0.205	0.752	4.571	38.316
	1.2E+46	3.2E+40	1.9E+36	1.8E+26	7.5E+16	1.26E+05	1.0E+00
	0.000056	0.000190	0.000480	0.002800	0.009500	0.033000	0.110000
	0.000180	0.000560	0.001400	0.007000	0.022000	0.075000	0.240000
0.500	$>10^{7}$	1500000	140000	9.40E+03	1.70E+03	460	270
	0.047	0.156	0.481	3.82E+00	16.872	60.463	130.562
	1.1E+37	1.7E+28	1.2E+20	1.36E+06	1.0E+00	1.0E+00	1.0E+00
	0.000250	0.000990	0.003300	0.016000	0.041000	0.100000	0.220000
	0.000790	0.002700	0.008000	0.036000	0.092000	0.220000	0.470000
0.600	3000000	81000	1.10E+04	1.20E+03	510	330	300
	0.21	0.92	3.79E+00	25.30	60.77	114.55	184.55
	1.5E+26	2.9E+15	1.46E+06	1.0E+00	1.0E+00	1.0E+00	1.0E+00
	0.006400	0.022000	0.050000	0.120000	0.200000	0.300000	bit err rate
	0.016000	0.051000	0.110000	0.260000	0.420000	0.620000	byte err rate
0.750	1.80E+04	1.70E+03	720	460	460	410	ave run to fail
	5.12E+00	24.4	54.7	108.7	148.0	205.4	mean RS corrs
	3.07E+04	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	max bytes

Ratio: (Not Constrained) / (Constrained)

	0.01	0.02	0.03	0.05	0.07	0.10	0.15
	n/a	n/a	n/a	n/a	1.3	1.3	0.8
	n/a	n/a	0.8	1.2	1.3	1.4	0.9
0.166	n/a	n/a	n/a	n/a	1.1	1.0	1.0
	1.0	0.9	1.0	3.1	1.0	1.0	1.0
	1.0	15.0	2.1	0.0	1.4	0.7	1.2
	n/a	0.8	1.0	1.1	0.9	1.0	0.8
	1.3	0.9	1.0	1.1	1.2	1.1	0.9
0.250	n/a	n/a	n/a	1.2	1.5	1.1	1.1
	1.1	0.9	0.8	1.0	0.8	1.0	0.9
	0.2	5.3	26.8	0.8	33.6	2.0	1.7
	0.7	0.8	0.9	0.9	0.9	0.8	0.8
	0.8	1.0	1.1	1.0	1.1	0.9	0.8
0.333	n/a	n/a	3.2	1.8	1.4	1.1	1.1
	0.9	0.9	1.0	0.9	0.9	0.9	0.9
	13.6	4.0	1.7	12.4	7.2	2.9	1.0
	1.0	0.9	1.0	0.9	0.8	0.7	0.7
	1.1	1.1	1.1	1.0	0.9	0.8	0.8
0.500	n/a	1.9	2.0	1.3	1.2	1.0	0.8
	1.1	1.1	1.0	0.9	0.8	0.9	1.0
	0.3	0.4	1.6	8.7	2.2	1.0	1.0
	0.6	0.8	0.7	0.7	0.7	0.7	0.8
	0.7	0.8	0.8	0.8	0.7	0.8	0.8
0.600	n/a	1.6	1.4	1.2	1.0	0.7	0.6
	0.7	0.8	0.8	0.8	0.9	1.1	1.1
	882.3	63.2	40.6	1.0	1.0	1.0	1.0
	0.3	0.5	0.5	0.6	0.7	0.8	bit err rate
	0.3	0.5	0.5	0.6	0.7	0.8	byte err rate
0.750	1.7	1.3	1.0	0.7	0.6	0.6	ave run to fail
	0.4	0.6	0.7	0.9	1.0	1.0	mean RS corrs
	4.8E+05	1.8	1.0	1.0	1.0	1.0	max bytes

Fig. S3. Left: Same as main text Figure 2, but for the case where DNA output constraints are imposed: No homopolymer runs > 4, $4 \leq$ CG ≤ 8 in any window of 12. Right: Ratio of unconstrained to constrained values for all quantities on the left. Generally the changes are insignificant. Only the three combinations of code-rate and error-rate in outlined boxes are significantly different, changing formerly marginal cases to now-infeasible ones. We conclude from these results and Figure S4 that the penalty on HEDGES in imposing the most common output constraints is small.

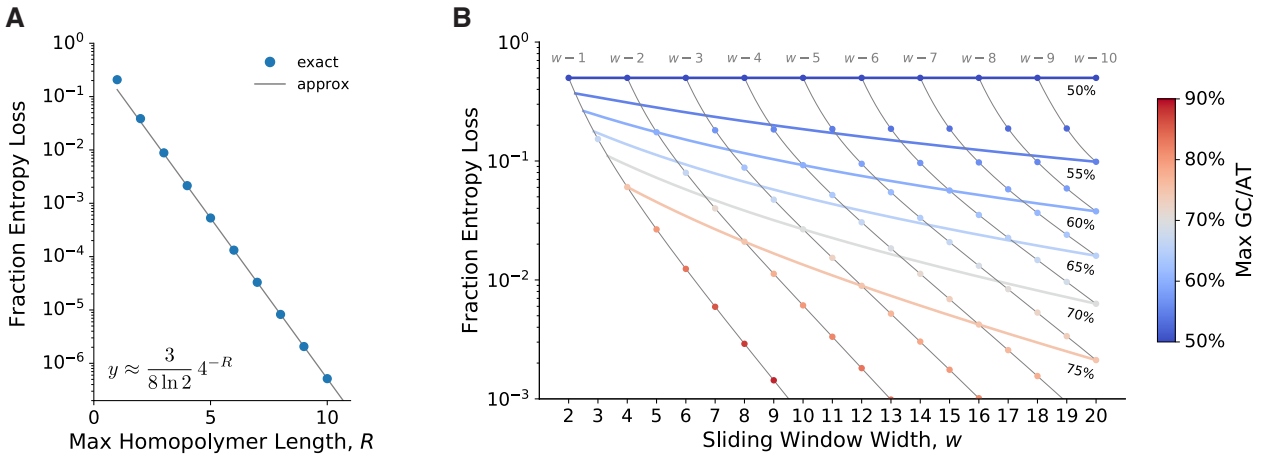

Fig. S4. Entropy loss from constraints. A) Entropy loss from homopolymer run constraints. B) Entropy loss from GC content constraints. GC content constraints are symmetric and are given by their maximal value, such that 40-60% GC is shown as 60% Max GC/AT. Colored lines give interpolated splines for a few GC content values of interest. Colored dots give realizable values in discrete DNA space. Grey lines give the integer value of the Max GC/AT nucleotides in terms of sliding window width, w. For example, for window length $w = 20$, 65% Max GC/AT is equivalently $w - 7 = 13$ Max GC/AT nucleotides. From (A) and (B), the two most common output constraints under reasonable parameters have combined entropy loss of only a few percent, as simulated in Figure S3.

code rate:	Total DNA error rate (with sub = ins = del): 0.01	0.02	0.03	0.05	0.07	0.10	0.15
	$<10^{-6}$	$<10^{-6}$	$<10^{-6}$	$<10^{-6}$	0.000002	0.000005	0.000016
	$<10^{-6}$	$<10^{-6}$	0.000001	0.000004	0.000007	0.000026	0.000084
0.166	$>10^{7}$	$>10^{7}$	$>10^{7}$	6000000	1900000	180000	8700
	0.0021	0.0021	0.0022	0.0043	0.0120	0.1151	2.2238
	1.3E+60	1.3E+60	3.8E+59	5.2E+54	1.3E+47	3.0E+30	2.97E+09
	0.000001	0.000003	0.000005	0.000015	0.000034	0.000140	0.000720
	0.000005	0.000015	0.000025	0.000065	0.000150	0.000570	0.002800
0.250	$>10^{7}$	$>10^{7}$	$>10^{7}$	2500000	420000	27000	1800
	0.0032	0.0057	0.0072	0.0240	0.0825	0.8499	11.0903
	7.1E+56	4.7E+52	7.0E+50	1.0E+42	8.3E+32	1.0E+16	1.26E+01
	0.000010	0.000025	0.000049	0.000150	0.000450	0.001800	0.009800
	0.000039	0.000099	0.000190	0.000550	0.001500	0.005500	0.027000
0.333	$>10^{7}$	$>10^{7}$	9900000	510000	67000	6800	680
	0.012	0.027	0.051	0.177	0.666	4.206	34.955
	1.7E+47	1.3E+41	3.3E+36	2.2E+27	5.4E+17	3.71E+05	1.0E+00
	0.000054	0.000180	0.000490	0.002400	0.007500	0.024000	0.082000
	0.000190	0.000620	0.001600	0.006700	0.020000	0.060000	0.190000
0.500	$>10^{7}$	2800000	280000	1.20E+04	2.10E+03	460	220
	0.051	0.165	0.467	3.259	14.313	56.681	136.191
	3.2E+36	6.6E+27	1.9E+20	1.2E+07	2.2E+00	1.0E+00	1.0E+00
	0.000160	0.000750	0.002300	0.011000	0.029000	0.073000	0.180000
	0.000540	0.002200	0.006400	0.027000	0.068000	0.170000	0.390000
0.600	$>10^{7}$	130000	1.50E+04	1.40E+03	490	240	190
	0.14	0.71	2.91	20.71	56.09	122.91	201.99
	1.3E+29	1.8E+17	6.0E+07	1.0E+00	1.0E+00	1.0E+00	1.0E+00
	0.002000	0.010000	0.025000	0.073000	0.130000	0.230000	bit err rate
	0.005400	0.024000	0.057000	0.160000	0.290000	0.480000	byte err rate
0.750	3.10E+04	2.20E+03	740	310	280	230	ave run to fail
	2.0	14.8	40.5	102.8	141.5	205.6	mean RS corrs
	1.5E+10	1.8E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	max bytes

Fig. S5. Measured in silico performance of the HEDGES algorithm, and inferred performance when concatenated with an interleaved outer RS(255,223) code. Each box is a different combination of code rate r and simulated DNA error rate P_{err}. Blue cells are measured values for HEDGES by itself: top value is bit error rate, middle is byte error rate, bottom is mean run of DNA characters to a decode failure. Red, yellow, and green cells assume a specific interleaved design (see text), and show mean equivalent correctable errors per RS decode (top value), and mean number of message bytes before an uncorrectable error (bottom value). Color indicates overall feasibility for error free data storage at petabyte or exabyte scale.

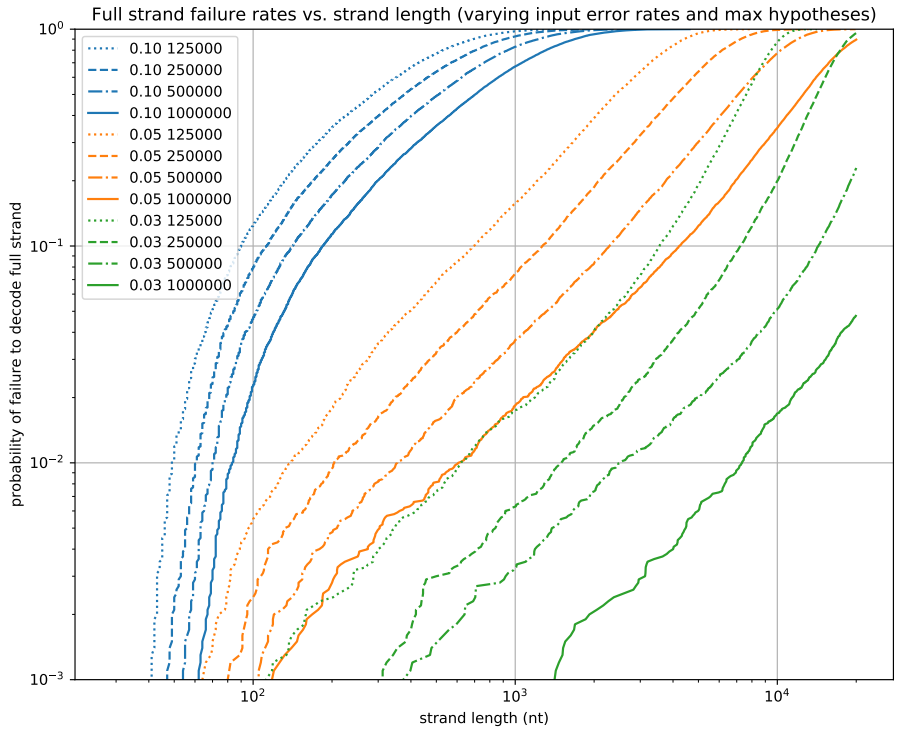

Fig. S6. HEDGES decode failure rates vs. strand length of a half-rate code as a function of total DNA error rate (3%, 5%, 10%) and hypothesis budget H_{limit} (1.25×10^5, 2.5×10^5, 5×10^5, 1×10^6).

References

1. Press WH, Teukolsky SA, Vetterling WT, Flannery BP, *Numerical Recipes: The Art of Scientific Computing*, Third Edition (Cambridge University Press, 2007), p. 352.
2. Hart PE, Nilsson NJ, Raphael B, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, SSC4, vol. 4 no. 2, pp. 100-107 (1968)