Automated design of oligopools and rapid analysis of massively parallel barcoded measurements

Ayaan Hossain[1], Daniel P. Cetnar[2], Travis L. La Fleur[2], James R. McLellan[2], and Howard M. Salis[1-4] *

[1]Bioinformatics and Genomics, [2]Department of Chemical Engineering, [3]Department of Biological Engineering, [4]Department of Biomedical Engineering, Pennsylvania State University, University Park, PA 16802, USA.

* Corresponding author: Howard M. Salis, salis@psu.edu

Abstract

Oligopool synthesis and next-generation sequencing enable the construction and characterization of large libraries of designed genetic parts and systems. As library sizes grow, it becomes computationally challenging to optimally design large numbers of primer binding sites, barcode sequences, and overlap regions to obtain efficient assemblies and precise measurements. We present the Oligopool Calculator, an end-to-end suite of algorithms and data structures, that rapidly designs many thousands of oligonucleotides within an oligopool and rapidly analyzes many billions of barcoded sequencing reads. We introduce several novel concepts that greatly increase the design and analysis throughput, including orthogonally symmetric barcode design, adaptive decision trees for primer design, a Scry barcode classifier, and efficient read packing. We demonstrate the Oligopool Calculator's capabilities across computational benchmarks and real-data projects, including the design of over four million highly unique and compact barcodes in 1.2 hours, the design of universal primer binding sites for one million 200-mer oligos in 15 minutes, and the analysis of about 500 million deep sequencing reads per hour, all on an 8-core desktop computer. Overall, the Oligopool Calculator accelerates the creative use of massively parallel experiments by eliminating the computational complexity of their design and analysis.

Keywords

automated design, massively parallel reporter assay, oligopools, barcodes, deep sequencing, genetic parts

Introduction

Massively parallel DNA oligonucleotide synthesis has transformed synthetic biology and gifted it with economies of scale *(1-3)*. Several commercially available services provide the custom synthesis of thousands of DNA oligonucleotides with researcher-defined sequences, mixed together in pools that are commonly known as oligopools. Oligopools are becoming more widely used as the starting DNA material to construct large libraries of genetic systems, while also facilitating the tracking and quantification of each genetic system's function through DNA barcoding. These technologies have already dramatically increased the scale, breadth, and precision of quantitative measurements, which have been leveraged across several applications to greatly push forward the state-of-the-art in Synthetic Biology *(4)*.

For example, synthetic biologists have leveraged oligopools to design and build large libraries of promoters *(5-7)*, hammerhead ribozymes *(8)*, biosensors *(9)*, genetic circuits *(10)*, protein scaffolds *(11)*, therapeutic molecules *(12)*, and RNA-based devices *(13)*, followed by characterizing their functions using massively parallel reporter assays (MPRAs). MPRAs combine sequence-encoded

readouts with next-generation sequencing (NGS) to quantify cell library compositions and measure mRNA & protein levels *(14-17)*. By enabling the construction and characterization of large numbers of genetic systems, the resulting datasets have been applied to determine the sequence determinants of several biological processes, including transcription initiation *(7, 18, 19)*, mRNA decay *(20)*, eukaryotic translation initiation *(21)*, and alternative transcript polyadenylation *(22)*. Large-scale CRISPR libraries have also been constructed using oligopool synthesis and screened using MPRAs *(23-25)*, while oligopool synthesis has been harnessed for storing information in DNA-based files *(26)*. As oligopool technology development has accelerated, the number and maximum length of oligonucleotides in an oligopool continues to increase, while the material costs of manufacturing oligopools has greatly decreased, mirroring the supra-exponential improvements in NGS technology. Thus, we expect that oligopools will continue to be leveraged to build large genetic system libraries, coupled to MPRAs for quantitative characterization.

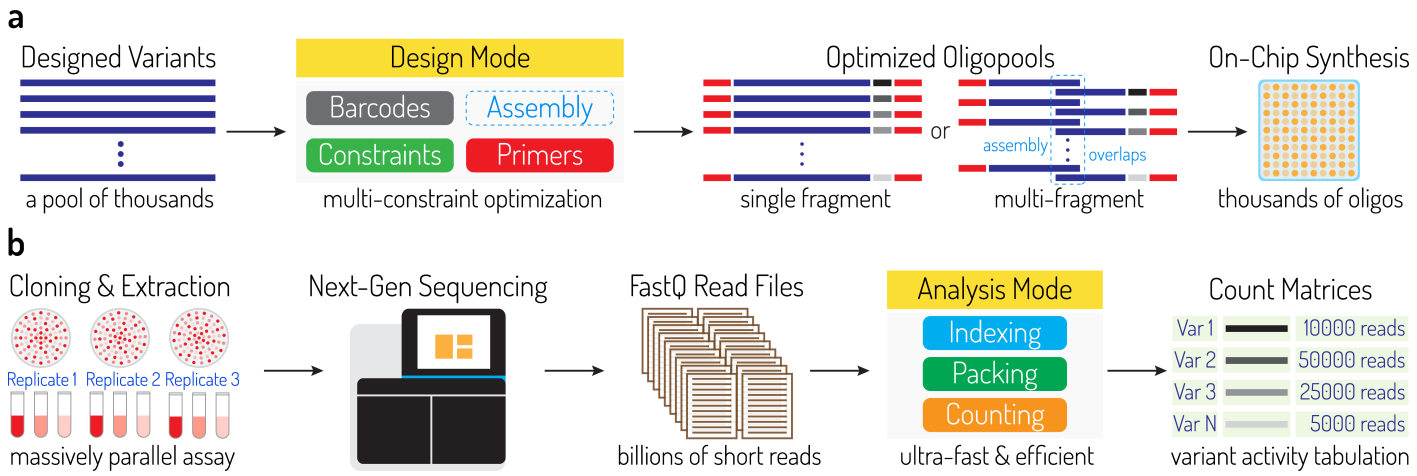

Figure 1. Design and analysis of oligopool variants using Oligopool Calculator. (**a**) In Design mode, oligonucleotide sequences are designed with optimized barcodes, primer binding sites, padding, and spacers. Longer constructs may also be split into shorter oligos for multi-fragment assembly. (**b**) In Analysis mode, barcode indexing, mapping, and counting are carried out on billions of reads to quantify variant frequencies, which are used to measure cell library composition and genetic system activities.

However, the design of oligopool sequences and their analysis after MPRA characterization remain a highly challenging and often bespoke effort across individual projects, motivating the development of a software pipeline that is specially designed to accelerate these efforts. While there are several algorithms for designing related parts and tasks, they were often developed with prior applications in mind and do not take into account the massive increase in throughput and sequence complexity present needed to design an oligopool and analyze its MPRA data. For example, IDT SciTools *(27)*, Primer3 *(28)*, and PrimerMapper *(29)* are well-developed software packages for designing pairs of PCR primers to amplify existing DNA templates. Primerize *(30, 31)* can split a single long construct into shorter oligo fragments. FreeBarcodes *(32)*, DNABarcodes *(33)*, and BARCOSEL *(34)* provide DNA barcode libraries and barcode sequence design algorithms *(35)*. SiteOut *(36)* designs spacer sequences without forbidden motifs. Kallisto *(37)*, HISAT2 *(38)*, and salmon *(39)* accurately map and count transcriptomic reads for studying differential gene expression. However, to date, there are no algorithms that can rapidly design ultra-specific PCR primer binding sites and primers across megabases of custom sequence, while incorporating sequence, structural, and model-based constraints needed for efficient PCR amplification, DNA assembly, and MPRA characterization. Existing barcode libraries and design

algorithms are decoupled from the overall oligopool design and analysis, though there are several workflow-wide constraints that must be satisfied, including ensuring minimum distinguishability criteria and the prevention of edge effects. Finally, the quantitative analysis of deep amplicon NGS libraries becomes computationally rate-limiting when files contain billions of reads across many thousands of barcoded variants. New algorithms are needed to rapidly design the many thousands of oligonucleotide sequences within an oligopool, while coupling both design and analysis tasks so that the resulting NGS data may be rapidly analyzed with high quantitative precision.

We introduce the Oligopool Calculator as a suite of design and analysis algorithms that enables the design of oligopool sequences (Design Mode) and the analysis of next-generation sequencing data from massively parallel reporter assays (Analysis Mode). In Design Mode, the algorithm designs barcodes, primer binding sites, primers, padding, spacers, and splits to facilitate the massively parallel PCR, DNA assembly, and cloning workflows used to build many thousands of genetic systems from a single oligopool (**Figure 1A**). Several workflow-wide criteria are mutually satisfied throughout the design of individual parts and the collective system. Barcode sequences are designed to be maximally distinguishable with a user-defined minimum pair-wise Hamming distance so that NGS base calling error does not lower the precision of the MPRA measurements. Primer binding sites and primers are designed for maximum specificity and efficiency during PCR, which includes precise matching of primer melting temperatures, eliminating all off-target binding, eliminating inhibitory DNA structures, and greatly reducing primer dimer formation. Padding sequences are designed to ensure uniform oligopool synthesis, PCR amplification, and DNA assembly. In multi-fragment mode, long DNA assemblies are automatically split into multiple oligonucleotide fragments with optimally selected overlap regions for *in vivo* homology-directed, Golden Gate, PCR, or Gibson assembly *(40-43)*.

In Analysis Mode, the Oligopool Calculator rapidly and efficiently carries out the analysis of deep amplicon sequencing libraries from MPRAs to quantitatively measure the frequency of each genetic system variant within the pooled library (**Figure 1B**). Analysis Mode is used to (i) index genetic system variants and their associated barcodes, (ii) filter, merge, and compress NGS reads into smaller read packs, and (iii) accurately map barcodes and count their occurrences from the packed read files. Both association and combinatorial counting are supported, enabling each genetic system variant to be associated with a single designed barcode sequence or a combination of multiple designed barcode sequences. Sequence matching is performed to quantify any deviation from expected sequences with reporting of mismatch distributions. The resulting count matrices are analyzed to measure genetic system activity across conditions of interest and can be combined with biophysics and/or machine learning to develop predictive sequence-to-function models. Altogether, in the following results, we demonstrate how the Oligopool Calculator was used across several recent MPRA projects, carry out computational run-time & memory benchmarks to show-case its extremely fast design & analysis throughput, and formally explain how the algorithms are able to deliver unprecedented speed and accuracy. The Oligopool Calculator is available at https://github.com/hsalis/SalisLabCode and https://github.com/ayaanhossain/oligopool under a GPLv3 open-source license.

Results and Discussion

Design of oligopools to build genetic part libraries

We begin with three concrete examples of how the Oligopool Calculator was used to design high-complexity oligopools containing several thousand oligonucleotides (**Figure 2**). In the first

example, we designed and constructed 93180 promoters with systematically varied sequence motifs, which was used to develop predictive models of transcription initiation rate, called the Promoter Calculator *(7)* and the Multi-Sigma Promoter Calculator (**Figure 2A**). The promoter variants are surrounded by several genetic parts to facilitate PCR, cloning, and characterization. The outer flanking regions contain an ultra-specific universal primer binding site, followed by flanking BamHI and HindIII restriction sites for digestion and ligation into a compatible vector. Next, the promoter variant is physically coupled to a unique barcode sequence. Barcode sequences were designed so that they could be uniquely distinguished even if they contained up to 6 mismatches or base calling errors. Finally, a pair of EcoRI and SpeI restriction sites facilitates a 2-step cloning procedure that enables the insertion of a long constant region in between the promoter variant and its corresponding barcode. In the final constructed systems, each designed promoter controls the transcription rate of a mRFP1 coding sequence with a designed ribosome binding site sequence that has a translation initiation rate of about 5000 au on the RBS Calculator v2.1 scale *(44-47)* (**Figure 2B**). Notably, the 2-step cloning procedure shifts the barcode position into the 3' untranslated region so that it cannot affect the transcription initiation rate.

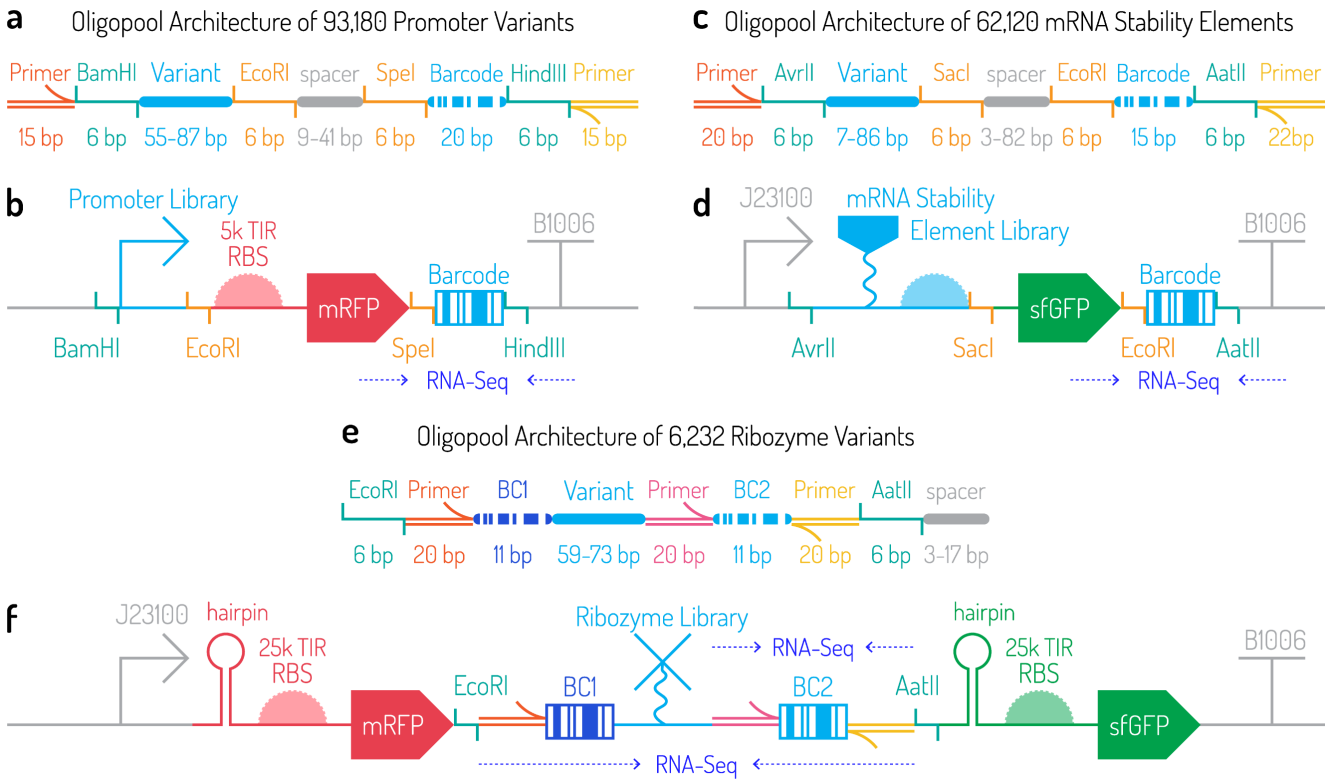

Figure 2. Demonstrative examples of oligopool design to build genetic part libraries. (**a**) An oligopool architecture with 93180 designed promoter variants and the (**b**) genetic system where promoters are inserted to measure their bacterial transcription initiation rates. (**c**) An oligopool architecture with 62120 mRNA stability elements and the (**d**) genetic system where sequences are inserted to measure their mRNA decay rate in bacteria. (**e**) An oligopool architecture with 6232 designed ribozymes, double barcodes, and triple primer binding sites and the (**f**) genetic system where ribozymes are inserted to measure their self-cleavage rate.

In the second example, we designed and constructed 62120 5' untranslated regions to systematically determine the interactions that control bacterial mRNA decay rates (**Figure 2C**). We used a similar oligopool architecture as in the first example, though the optimal PCR primer binding sites were longer, while the barcode sequences were shorter, all while satisfying optimal design criteria.

The restriction sites and enzymes used for the 2-step cloning procedure were different, though the Oligopool Calculator was able to rapidly design the needed primer binding sites, barcodes, and spacers, taking into account these changes. In the final constructed systems, the mRNA stability elements were introduced downstream of a constant promoter and followed by a sfGFP coding sequence (**Figure 2D**).

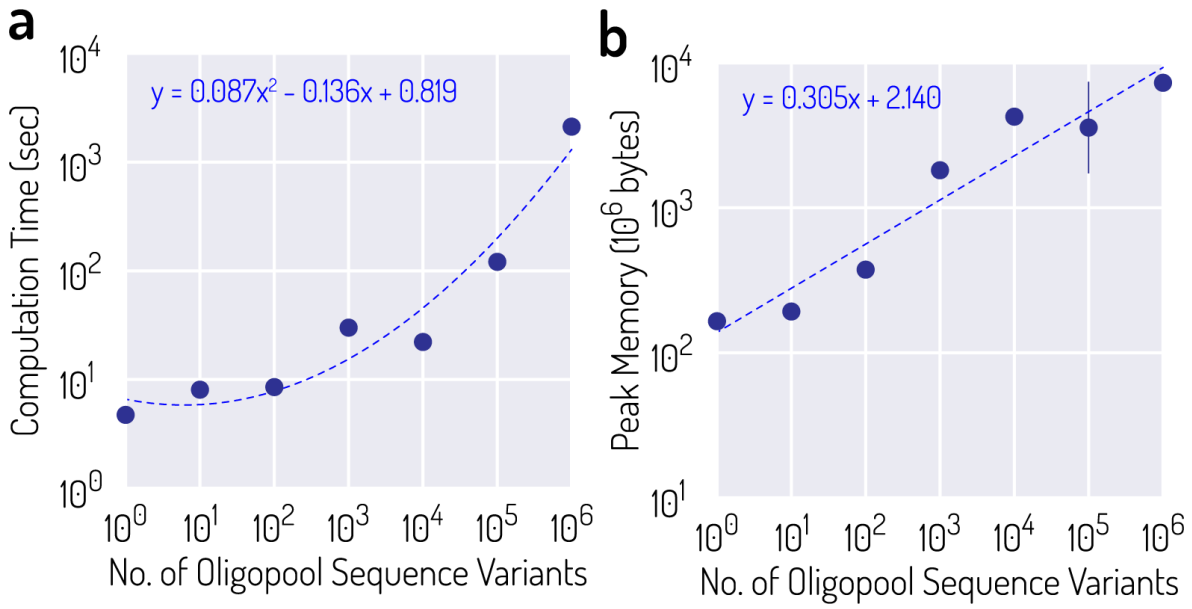

Figure 3. Oligopool Calculator Design Mode Benchmarks. (a) The computational runtime of designing an oligopool containing two primer binding sites, one barcode, and one spacer, using a single 3.5 GHz Intel processor on a desktop computer with 64 GB RAM, showing that runtime scales quadratically. **(b)** The computational memory (RAM) needed to design an oligopool scales linearly with oligopool size. Dots and bars are the mean and standard deviation of 10 independent runs.

In the third example, we designed and constructed 6232 highly non-repetitive hammerhead ribozymes, which increase mRNA stabilities and act as insulators within engineered genetic systems with many genetic parts (**Figure 2E**). The ribozymes were designed using Non-Repetitive Parts Calculator *(5)* to have a maximum shared repeat length of only 16 nucleotides, which enables them to be all be used simultaneously without inadvertently triggering homologous recombination. We used a 2-barcode strategy to measure the ribozymes' self-cleavage efficiencies, preparing two different barcoded amplicon libraries to quantify the amounts of cleaved versus full-length mRNAs. Here, the BC2 barcode counts are proportional to the total number of transcribed mRNAs whereas the BC1 barcode counts are proportional to the number of full-length (uncleaved) mRNA. By incorporating internal controls into the part library, these barcode counts can be normalized and compared, yielding the ribozymes' self-cleavage efficiencies. This application illustrates how multiple barcode sequences can be incorporated into an oligopool to quantify genetic part activities. Importantly, the Oligopool Calculator's analysis mode can track and remove artifacts that occur when longer amplicons are sequenced, using its multi-barcode combinatorial mapping. For example, when sequencing long amplicons, a small percentage of reads will contain mis-paired barcodes (called "index hopping"), which are identified during mapping and removed during counting.

More generally, the Oligopool Calculator is highly versatile and can be applied to design oligopools and analyze deep sequencing reads with diverse architectures. For example, combinatorial genetics en mass (CombiGEM) uses combinatorial DNA assembly to construct large toolboxes of genetic designs or screens, utilizing tandem arrays of DNA barcodes to uniquely identify each design *(48-50)*. The Oligopool Calculator can design tandem barcode arrays and analyze multi-barcode reads,

enabling tracking and quantification of genetic designs for studying gene regulation, biosensor development, or metabolic pathway engineering *(51-53)*.

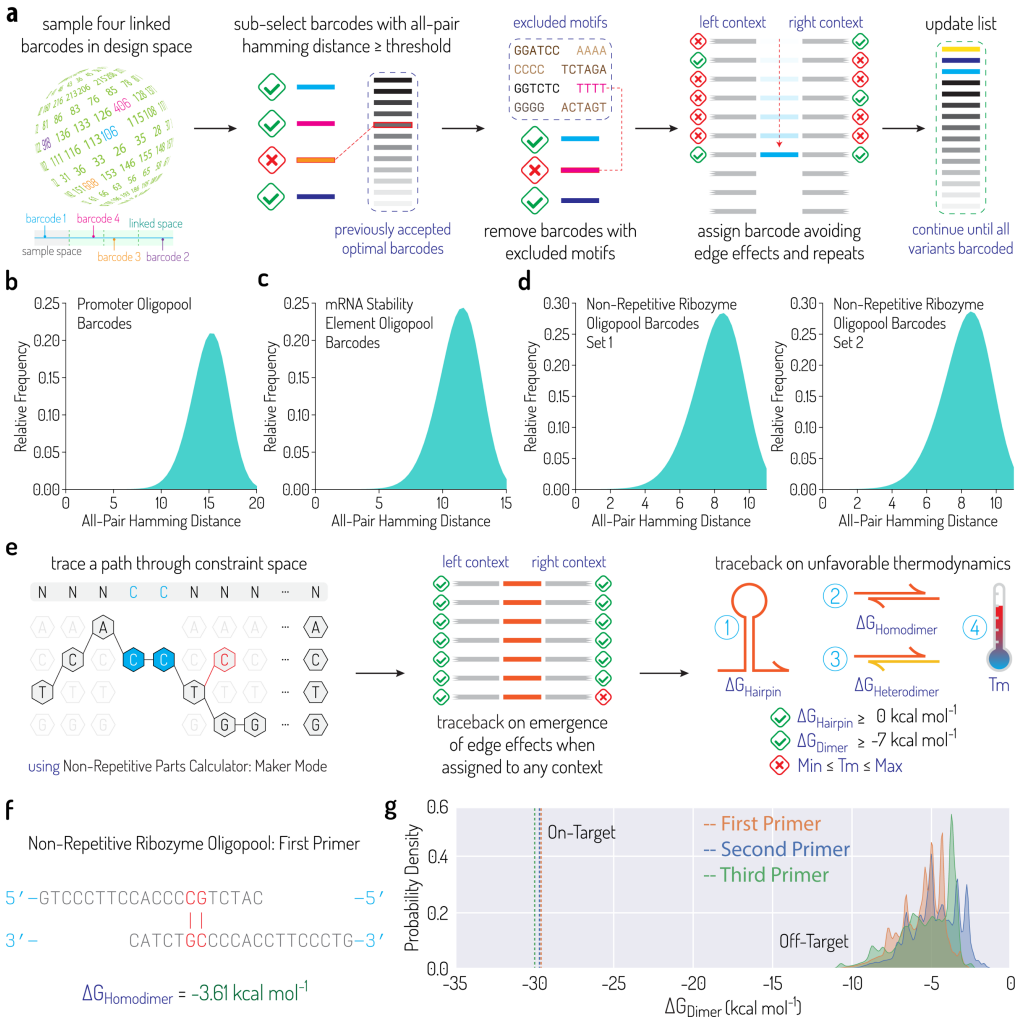

Figure 4. Barcode and primer design algorithms. (**a**) Barcode design algorithm schematic showing the selection of initial candidate set of four linked barcodes, and filtering them based on Hamming distance, excluded motifs, contextual non-repetitiveness, and edge effects. All pair barcode Hamming distance distribution for (**b**) promoter oligopool, (**c**) mRNA stability element oligopool, and (**d**) non-repetitive ribozyme oligopool barcode sets. (**e**) Primer design algorithm schematic showing the use of Non-Repetitive Parts Calculator to design primers based on a desired sequence constraint that is free of excluded motifs and edge effects when assigned to any context, and possess a desired level of non-repetitiveness between the primer, the oligopool and background sequences. Designed primers also do not form stable hairpins, or strong duplexes between paired primers and have a desired melting temperature. Examples of weak (**f**) homodimer formation between two copies of the first forward primer, and (**g**) a comparison between on-target primer binding and the distribution of off-target primer binding for the non-repetitive ribozyme oligopool architecture.

Optimal design of barcodes, primer binding sites, spacers, and overlaps

The Oligopool Calculator's design mode carries out custom sequence design of barcodes, primer binding sites, primers, spacers, and overlap regions using newly developed algorithms that generate solutions that satisfy all workflow-wide constraints while being extremely fast and memory-efficient. For example, designing an oligopool with one million sequence variants takes about 35 minutes on a typical desktop computer, depending on the oligopool architecture (**Figure 3**). Below, we describe each design algorithm and measure their effectiveness via several benchmarking tasks.

Barcodes Barcodes in an oligopool are used to uniquely identify each variant and quantify each variants' frequency in a sample. Barcodes are critical to properly tracking variants, especially when the variants have similar sequences. There are four key criteria to designing barcodes. First, there must be at least one barcode per variant, which dictates the number of barcodes that must be designed for the oligopool. Second, barcode sequences must be uniquely mappable with a tunable level of distinguishability as determined by the expected base calling error during next-generation sequencing. Distinguishability is quantified globally by the minimum pair-wise Hamming distance across all barcode sequences. Hamming distance is the number of mismatched nucleotides between two pairs of same-length barcodes. Third, the barcode length must be long enough to satisfy the first two design criteria, but not any longer, as oligonucleotides lengths are currently limited by synthesis processivity and space on the oligonucleotide remains scarce. Finally, all barcode sequences must not contain homopolymer tracts that are challenging to sequence, while also excluding a set of researcher-defined sequences, such as restriction sites, that are necessary to facilitate cloning. The barcode design algorithm generates an optimal set of barcode sequences that satisfy all four design criteria, custom tailored to the oligopool library and a researcher-defined level of minimum distinguishability (**Methods, Figure 4A**). For example, when sequencing an oligopool with 10-nt barcode sequences using Illumina NovaSeq, we expect a base calling error of 0.1%, yielding 0.991% of barcodes with a 1-nt mutation and 0.004% of barcodes with a 2-nt mutation *(54)*. A minimum pair-wise Hamming distance (H_{min}) of 2 would uniquely distinguish 99.996% barcode reads. However, when sequencing the same oligopool using Oxford nanopore sequencing, we expect a base calling error of 0.5%, yielding 4.8% of barcodes with a 1-nt mutation, 0.1% of barcodes with a 2-nt mutation, and 0.001% of barcodes with a 3-nt mutation *(55)*. Here, a H_{min} of 2 would uniquely distinguish 99.9% of barcode reads, while a H_{min} of 3 would improve distinguishability to 99.999%. Designing a larger number of barcodes with higher minimum pair-wise Hamming distances is a challenging computational task, especially when constrained to shorter barcode sequences.

Barcode design takes place in a transformed integer space where each barcode sequence of length L is represented by an integer n $(0 \geq n > 4^L)$. In a loop, we generate M barcode sequence candidates by splitting the integer space into M partitions, randomly selecting an integer from the first partition and leveraging the M-way symmetry of the design space to select $M-1$ additional DNA barcodes that are highly distinguishable. We then decode the integers as base-4 numbers and convert them back to DNA sequences. For example, if $L = 5$ and $M = 4$, the integer space spans 0 to 1023 and the partitions are 256 integers each. The initial random selection is $x_1 = 198$, which controls the selection of integers in the remaining 4 partitions, according to $x_2 = 4^L - 1 - x_1 = 825$, $x_3 = 4^L / 2 + x_1 = 710$, and $x_4 = 4^L / 2 - 1 - x_1 = 313$. These integers when converted to DNA sequences are ATACG, TATGC, GTACG, and CATGC, showing the symmetry of dispersed selection in reverse

compliment space. This approach enables very rapid generation of candidate barcode sequences even when L and M are large. In the next few steps, barcode candidates are removed if they do not satisfy all design criteria. The first filter is to remove any barcode candidate that contains excluded sequences, such as restriction cut sites or polymeric runs. The second filter is to efficiently calculate the pair-wise Hamming distances across all barcode candidates, using multi-threaded array broadcasting techniques *(56)*, and then remove any barcode candidate that does not meet the H_{min} threshold. Finally, we consider the potential edge effects of placing a candidate barcode adjacent to an upstream or downstream sequence within its designated position within an oligonucleotide (**Supplementary Figure 1**). We loop through each oligonucleotide in the oligopool and assign a barcode sequence to each position only if the edge effects do not invalidate the design criteria. Barcode generation continues until all oligonucleotides have an assigned barcode sequence that satisfies the four design criteria. Altogether, this barcode design algorithm saturates the k-mer spectrum and ensures that the terminal ends of each barcode have maximally dissimilar sequence.

Importantly, the algorithm is especially efficient at reducing the computational run time for the most intensive task of the design process, which is ensuring a minimum pair-wise Hamming distance across all barcode sequences. First, the symmetry of the candidate barcode selection algorithm ensures that every initially generated barcode sequence is about $(L - M)$ Hamming distance apart. Second, it is straight-forward to track the previously generated candidate barcodes and ensure the selection of new candidate barcodes to preclude double-work. Third, the pair-wise Hamming distances only need to be calculated for the newly generated candidate barcodes. In practice, the algorithm completes very quickly, taking less than 10 minutes for each of the three oligopool examples (**Figure 2**, **Supplementary Table 1**). The distributions of the pair-wise Hamming distances (**Figure 4BCD**) as well as the distributions of the pair-wise edit distances (**Supplementary Figures 2, 3, 4**) for these designed barcodes met and greatly exceeded the design criteria. The 20-bp barcodes had about 15 mismatches on average, the 15-bp barcodes had about 11 mismatches on average, and the 11-bp barcodes had about 8 mismatches on average. Importantly, the lowest value of the pair-wise Hamming distance in the distribution always exceeded or met the researcher-defined minimum value (1 to 3, depending on the oligopool design). At the extremes, the algorithm only needed 72 minutes to design 4194304 14-nt barcode sequences with a H_{min} of 2; when increasing the H_{min} to 3, the same barcode design procedure required 523 minutes (**Supplementary Table 1**).

Primer binding sites PCR is routinely used to selectively amplify oligopools, converting the mixture of single-stranded DNA oligonucleotides into double-stranded DNA fragments. However, PCRs on oligopool templates are more challenging and require more careful design of the primers and primer binding sites to avoid mis-priming, low PCR efficiency, and primer dimer formation. The Oligopool Calculator generates pairs of primers and primer binding sites with target length $(14 > L > 23)$ that satisfy 7 important design criteria. Designed sequences must not contain any researcher-defined excluded sequences, such as restriction sites and homopolymeric tracts, including edge effects (rule #1). Pairs of primers must have matching DNA melting temperatures (e.g. with a difference smaller than 0.3°C) within a research-defined range (rule #2). Primers must not strongly bind to any undesired sites on the DNA template, which is achievable by ensuring that all primer sequences have more than $(L - 10)$ mismatches to any potential binding site (rule #3). Single-stranded primers must not fold into any stable secondary structure $(\Delta G_{fold} \geq 0\ kcal/mol)$ that would inhibit template binding and PCR (rule #4). The 3' ends of both primer sequences must be either an A or a T to weaken primer dimer structures that could result in strand extension and PCR amplification (rule #5). Finally, the two primers must not

bind to each other and form competitively stable structures ($\Delta G_{dimer} \geq -7\ kcal/mol$), including homodimer and heterodimer structures (**Supplementary Figures 5, 6, 7**), that would again inhibit template binding and PCR (rules #6, #7).

While some of these design criteria are well-known to the field, the Oligopool Calculator combines model-based path generation and k-mer hash tables to very rapidly generate primer solutions even when oligopools are complex. Pairs of universal primers were designed for each of our real-world oligopool examples (**Figure 1**) in fewer than 30 seconds (**Supplementary Table 1**), satisfying all design rules. At the extreme, the Oligopool Calculator only required 816 seconds to design a pair of universal primers for a million 200-mer oligonucleotides (equivalent to 200 Mbp of template DNA) (**Supplementary Table 1**).

To do this, we leverage our previously developed Non-Repetitive Parts Calculator algorithm *(5)*, which designs large toolboxes of genetic parts that do not contain any k-mers that match to either another genetic part in the toolbox or to an inputted "background" sequence. Here, the background sequence is the list of oligonucleotides in the oligopool DNA template (before the primer binding sites are added) as well as any other input template DNA sequence that must be avoided (e.g. the genome sequence of *E. coli*, which can appear as contaminant DNA in many labs). The Non-Repetitive Parts Calculator accepts sequence, structure, and function constraints, which are applied during path generation to avoid excess loss-of-work. We then use the Non-Repetitive Parts Calculator to generate a large number of candidate primer sequences with a maximum repeat length of 10 nucleotides, while applying sequence constraints to ensure rule #5, single-stranded DNA structure constraints to satisfy rule #4, and a string-matching function to ensure rule #1. The resulting toolbox of candidate primers satisfy rule #3 by construction. We then calculate the DNA melting temperatures of each candidate primer and pair primers together when they satisfy rule #2. For every candidate pair of primers, we use ViennaRNA *(57)* to calculate their homodimer and heterodimer minimum-free-energy structures, discarding any primer pairs that break rules #6 and #7. This procedure continues until at least one pair of primers has been generated or until a counter has been exceeded. These calculations are carefully ordered so that less expensive operations (e.g. k-mer collisions) take place before more expensive operations (e.g. calculating DNA structures) during path generation and tracebacks (**Methods**, **Figure 4E**). The sequences and orientations of the designed primers are then used to determine their corresponding primer binding sites within the oligopool. This design procedure results in highly specific primers with extremely low off-target binding to the oligopool template (**Figure 4G**, **Supplementary Figure 8**).

Restriction sites, motifs and spacers The Oligopool Calculator's design mode can introduce restriction sites, researcher-defined sequence motifs, and spacer regions into the oligopool design, while ensuring that they maximally satisfy the workflow-wide design rules. These sequences are defined using the IUPAC degenerate nucleotide code with an option to set the length of a spacer region to an oligonucleotide-specific value, making them useful as padding elements or as anchor sequences used to define barcode positions. During sequence design, the Oligopool Calculator's pathfinding algorithm evaluates the input specifications and identifies an oligonucleotide-specific sequence solution that satisfies design rule #1, preventing the appearance of researcher-defined excluded motifs and homopolymer tracts (including edge effects from oligonucleotide-specific flanking sequences). The design process is very fast; for example, designing the spacer regions for 93180 oligonucleotides required only 6.8 seconds (**Table S1**).

Splits and overlap regions While maximum oligonucleotide synthesis lengths continue to improve, there will always be genetic designs that are too long to fit within one oligonucleotide. One solution is to split the genetic design into multiple overlapping oligonucleotides and apply either *in vitro* enzyme assembly or *in vivo* recombination to build the full genetic system. The Oligopool Calculator facilitates this process by selecting the optimal split points and overlap regions within a long genetic system design with the overall goals of maximizing the sequence-specificity of the overlap regions, while ensuring that the overlap regions have a DNA melting temperature above a minimum setpoint. Both the DNA melting temperature and sequence specificity of the overlap region are important when using any DNA assembly technique that uses many Watson-Crick base pairings to determine assembly order and ligation fidelity, which includes *in vitro* PCR assembly, *in vitro* Gibson assembly, and *in vivo* homologous recombination. For example, we previously applied the Oligopool Calculator to build 1722 yeast promoters with a 325-bp variable region by splitting each design into a pair of designed 200-nt oligonucleotides with a 20-nt overlap region, yielding an oligopool of over 3444 oligonucleotides *(5)*.

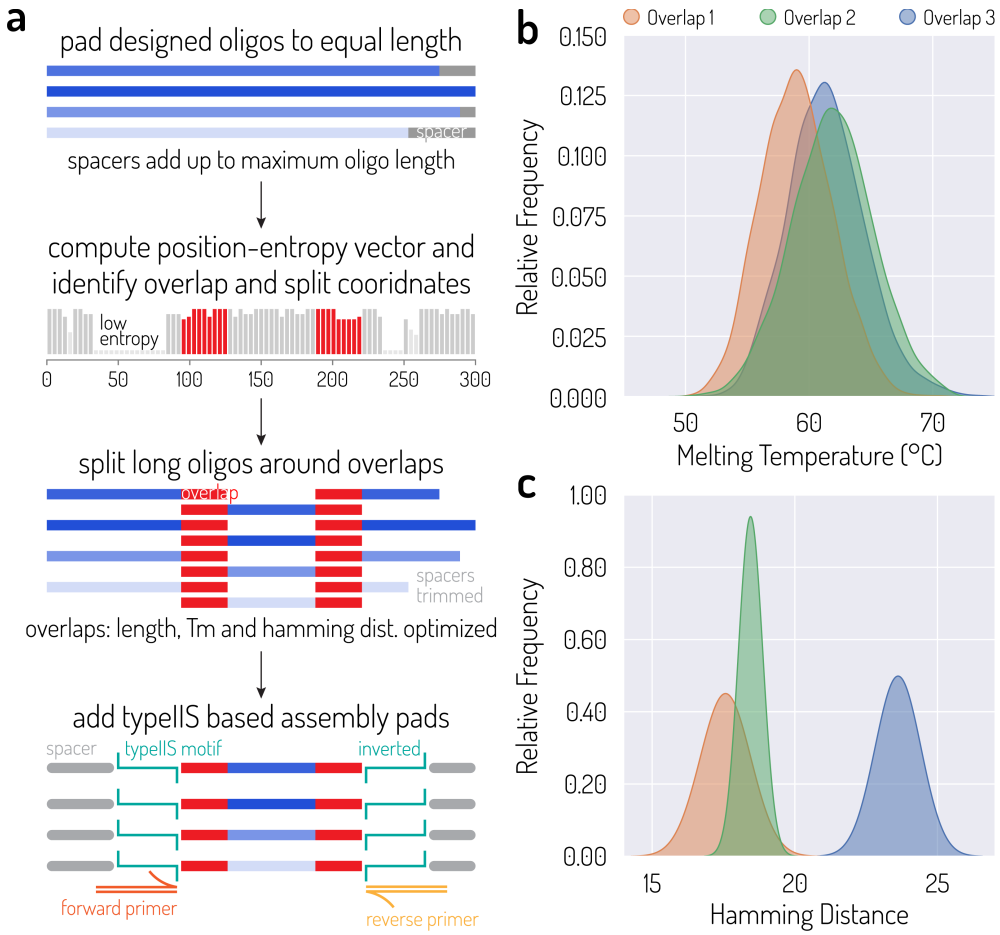

Figure 5. Oligopool splitting and padding. (**a**) The Oligopool Calculator identifies optimal split sets and overlap regions for multi-fragment homology-based assembly. Overlap regions must have a minimum melting temperature and minimum pairwise Hamming distances. PCR primer binding sites, cut motifs, and spacers are then added to split regions to create oligonucleotides and oligopools. (**b**) The distributions of DNA melting temperatures are shown for three sets of designed overlap regions, satisfying a minimum DNA melting temperature of 50°C. (**c**) The pair-wise Hamming distance distributions are shown for three sets of designed overlap regions, showing high sequence variability between designed overlap regions.

The overlap regions all had DNA melting temperatures above 50°C. In this example, DNA assembly was carried out via *in vitro* one-pot PCR assembly, followed by genome integration via multi-fragment *in vivo* recombination inside yeast cells with both positive and negative selection.

To do this, the Oligopool Calculator first carries out a left-alignment of all the genetic designs, adding padding sequence to the 3' terminal ends where necessary to achieve uniformity, followed by calculating the number of oligonucleotides needed to build them, according to researcher-defined oligo specifications (**Methods, Figure 5A**). The algorithm then calculates the Shannon entropy for each aligned position along the genetic design axis to identify the sequence regions that have the highest variability. The algorithm then selects a set of optimal sequence regions that have maximal lengths (limited by the maximum oligonucleotide length) with overlap sequences that have maximal entropy. Once the optimal set of splits are defined, the overlap region length is optimized. The overlap region length is varied uniformly for all assemblies to ensure that the overlap regions' DNA melting temperatures are above a researcher-defined setpoint as well as to ensure that all pairs of overlap regions have Hamming distances above a minimum threshold. If the overlap region length exceeds a threshold (defined by the maximum oligonucleotide length), then the next most optimal set of splits is selected. Once this procedure finishes, the oligonucleotides are designed. If any terminal padding sequences were added during the alignment of the genetic system designs, then they are removed. Flanking restriction sites, motifs, and PCR primer binding sites are then added as described previously to facilitate PCR amplification and removing the PCR primer binding sites from the amplified DNA fragments. As another demonstration, we reran the Oligopool Calculator on the 1722 yeast promoter regions and performed a 4-way splitting of the genetic system designs into an oligopool with a maximum region length of 134 bp and using a minimum DNA melting temperature of 50°C for the three overlap regions. The resulting split sets and overlap regions were found to have high DNA melting temperatures (**Figure 5B**) and with high pair-wise Hamming distances (**Figure 5C**), satisfying the design rules for sequence-specific assembly. However, in order for this oligo splitting strategy to be successful, the genetic designs must have sufficiently long sequence regions with high sequence diversity, for example, by introducing non-repetitive genetic parts or barcode sequences into the designs.

Rapid analysis of massively parallel measurements on barcoded genetic systems

While the Oligopool Calculator's Design Mode carries out the automated design of oligopools to facilitate the construction of barcoded genetic systems, its Analysis Mode is responsible for rapidly analyzing next-generation sequencing data that measures the functions of those barcoded genetic systems. For example, an oligopool is designed, synthesized, and utilized to build large library of plasmids that uses different regulatory or coding genetic parts to alter cell function (**Figure 2**). Barcodes are strategically positioned to provide a quantitative proxy measurement, which can include library composition, mRNA levels, or another coupled cell function (e.g. using a cell sensor, cell sorting, or growth-coupled enrichment). Cells are transformed with the plasmid library, grown in desired environmental conditions, and sampled across kinetic timepoints or during steady-state. The DNA and mRNA from those cell samples are then processed and sequenced, using short-read sequencing or long-read sequencing, and yielding millions to billions of barcode-containing reads. The Oligopool Calculator analyzes these read files and very rapidly determines the barcode counts for each genetic system variant in the library, while reducing the effects of confounding factors that could alter barcode counts (e.g. assembly mutations or base-calling errors).

To do this, the Oligopool Calculator carries out an automated procedure that includes: (i) automated training of an extremely robust barcode classifier, using a Scry machine learning model to quickly and accurately identify barcodes even if the sequence contains several mutations or base-calling errors; (ii) automated filtering, merging, and packing of reads, which greatly reduces storage

sizes; and (iii) rapid counting of barcodes (in association or combinatorial modes) with the option of running barcode-dependent call-back functions on the genetic system's sequence. Overall, Analysis Mode is extremely fast with throughputs of 180 to 720 million reads per hour on a typical 8-core desktop computer, depending on the sequencing format and number of barcodes (**Table S3, Table S4**).

Barcode Mapping To begin, the researcher inputs a list of expected (reference) barcode sequences for each barcoded position in the genetic system library as well as an anchor sequence that defines the locations of the barcoded positions. The anchor sequence is any constant sequence that has a constant distance to the barcode sequence, enabling rapid barcode mapping and counting. The anchor sequence does not need to be adjacent to the barcode, but it must be located within the same read during next-generation sequencing. The Oligopool Calculator then analyzes the reference barcodes and automatically generates a list of all uniquely identifiable barcode sequences, including mismatches. These sequences are then stored in an efficient look-up table (**Methods, Figure 6A**). The maximum pair-wise Hamming distance across the reference barcodes is then calculated and used as the default tolerance for mismatches. Any non-unique reference barcodes are reported to the researcher.

As a key innovation, we greatly accelerate barcode mapping by developing a novel classifier (called Scry) to rapidly & accurately predict the barcode label from an imperfect read sequence, including read sequences that contain several mismatches or indel mutations. Scry is a k-nearest neighbor classifier that uses the position-ordered k-mer motifs within an extracted barcode sequence to quickly identify its reference barcode sequence, while being robust to multiple mismatches or indel mutations. For example, with $k = 5$, there are 16 ordered 5-mers in a 20-nt barcode sequence. If any two nucleotides in the read sequence are mutated or deleted, then anywhere from 6 to 10 of the 5-mers will be mutated, leaving 6 to 10 unaffected 5-mers as identifiers. The Scry classifier uses a rapid matching process to identify such unaffected *k*-mers and then utilizes them to uniquely identify the reference barcode or reject the sequence as non-identifiable (**Methods**).

In greater detail, the Scry classifier first checks whether the barcode sequence extracted from a read is an exact match to a reference barcode sequence. If there is no exact match, the classifier decomposes the extracted barcode sequence into positioned k-mers, followed by carrying out a scored matching process that compares the positioned k-mers of the extracted barcode sequence to the positioned k-mers of all the reference barcode sequences. A match is favorably scored whenever matching k-mers have the same order of appearance, but not if matching k-mers appear out-of-order. The extracted barcode sequence is labeled as potentially arising from a reference barcode if it has the highest scoring match. This matching process is much faster than traditional gapped sequence alignment algorithms, particularly when (in our case) there are tens of thousands of reference sequences mapped across billions of reads, while also being memory efficient. From the scored match process, the Scry classifier identifies a small set of reference barcode sequences that potentially match to the extracted barcode sequence, greatly simplifying the identifiability challenge. The Scry classifier then identifies the highest matching reference barcode by carrying out a small number of pair-wise alignments and calculating their edit distances, which is the total number of mismatches or indel mutations needed to convert one sequence to the other. The Scry classifier then uses the reference barcode sequence with the smallest edit distance to label the extracted barcode sequence. Importantly, the Scry classifier is robust to the presence of multiple mismatches or indel mutations without losing

identifiability; however, if there are excess mutations, multiple reference barcodes will all have the same

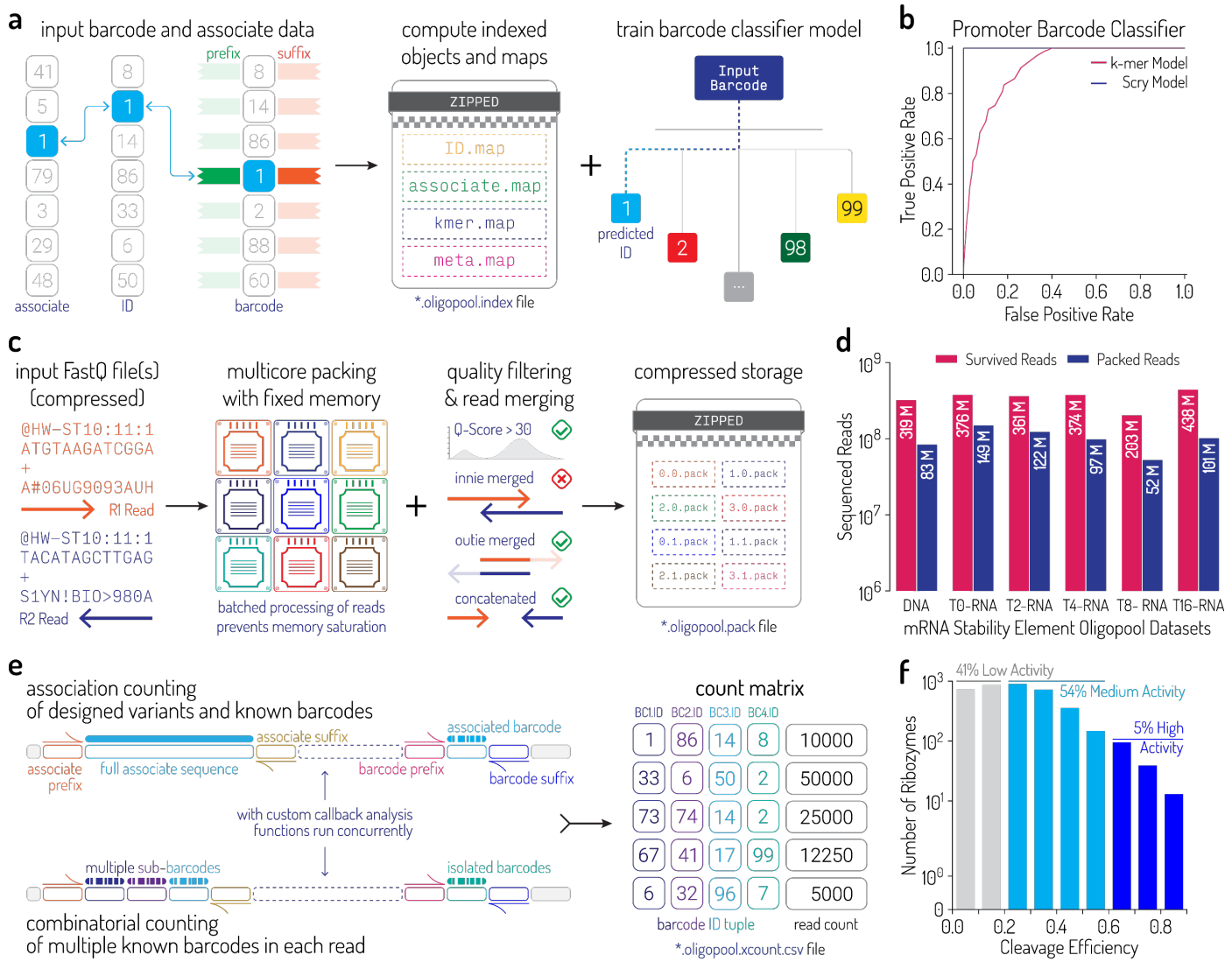

Figure 6. Indexing, packing, and counting of sequenced reads from oligopool characterization experiments. (**a**) Algorithm schematic for computing efficient indexes of barcodes, and associated variants, and training an accurate barcode classifier model. (**b**) Scry barcode classifier test results (blue lines) for the σ^{70} bacterial promoter oligopool barcodes show a mean of 99% area under the ROC curve score, compared to a baseline mean score of 91% (red lines). A total of 100 test cycles were conducted, with 10,000 barcodes tested in each iteration. (**c**) Schematic showing multicore packing of sequenced reads after filtering reads based on quality scores and read length, and any optional merging or concatenation of paired reads. Each read pack then contains processed reads along with frequency counts of their occurrence within a batch. (**d**) Bar chart showing total number of amplicons samples sequenced at various time points (red bars) to characterize the mRNA stability element oligopool, and the total number of unique reads after packing them efficiently (blue bars). (**e**) Algorithm schematic showing association, and combinatorial counting of barcoded reads to produce count matrices. In association counting, the sequenced barcode is identified, and its association with the designed variant is verified. In combinatorial counting, multiple isolated barcodes, or various sub-barcode combinations in the reads are mapped and counted. (**f**) Bar chart showing the cleavage efficiency distribution of the non-repetitive ribozyme library from sequencing characterization experiment.

edit distance, enabling the Scry classifier to label the extracted barcode sequence as non-unique.

We assessed the performance of the Scry classifier by carrying out barcode mapping on randomly mutated barcode sequences to determine how well they could be correctly labeled. We introduced between 2 to 6 random mutations (substitutions, insertions, or deletions) at random positions across 10000 different reference barcode sequences and then applied Scry to predict the original reference barcode. We then repeated this test 100 times to determine the true and false positive rates. Overall, the average area under the receiver operating characteristic curve (AUROC) score for Scry barcode classification was over 99%, compared to an AUROC of 91% for a baseline classifier that only uses unordered k-mers (**Figure 6B**). Similarly, in our real-world examples, the Scry model had AUROC scores of 98% to 99% when carrying out barcode mapping on our example applications, as compared to 92% to 93% when using the baseline classifier (**Supplementary Figure 9**). The Scry classifier is computationally very fast. Index initialization only required about 0.2 seconds for 6232 reference barcodes and about 2 seconds for 62120 reference barcodes (**Table S2**). Barcode mapping with Scry has an exceptionally high throughput of over 1 billion reads per hour (**Table S4**) even when there are many thousands of reference barcodes.

Read Merging, Filtering, and Packing The next key step is the processing of the next-generation sequencing reads, including read merging, filtering, and packing (**Figure 6C**). While read merging and filtering are commonly carried out *(58-62)*, read packing is a simple, but novel, technique that greatly reduces computational storage requirements and accelerates the barcode mapping and counting process. First, the Oligopool Calculator assesses the reads in FastQ format for base calling accuracy (Q-score) and discards any reads with Q-scores less than a researcher-defined threshold. Reads that are substantially shorter than an expected researcher-defined length are also discarded. Next, the Oligopool Calculator automatically determines the best read merging parameters for paired-end reads, which may be over-ridden by researcher-defined options. The read merging parameters include merge orientation, minimum overlap length, and the maximum number of mismatches expected in the overlap region. For example, when reads have a strand orientation, paired-end reads can be merged in either the innie orientation (the overlap region is located on the 3' ends of the reads) or the outie orientation (the overlap region is located on the 5' ends of the reads). The overlap region's length depends on the amplicon architecture. If no overlap exists, the paired-end reads may be concatenated. The automated procedure samples 50000 reads from the dataset and determines the best merge parameters by counting successes across varied parameters and using their ranking for prioritization. The ranked list of parameters are then used to carry out read merging for the entire dataset. For example, the Oligopool Calculator begins by using the highest ranked parameters to carry out read merging, but if unsuccessful, it then repeats the process using the next highest ranked parameters, until success is achieved. This prioritized approach greatly reduces the computational cost of read merging. However, if the provided FastQ file already contains filtered and merged reads, the Oligopool Calculator provides an option to skip these steps.

Finally, the Oligopool Calculator carries out read packing to greatly accelerate the barcode mapping and counting process. When carrying out massively parallel experiments, we often collect next-generation sequencing results with very high read depths in order to obtain precise measurements. For example, one sequence variant might be read over 100000 times whereas another sequence variant might only be read 100 times. Overall, there will be many more reads than sequence variants even when considering that the number of unique variants will be much larger when including mutations and/or base calling errors commonly found in read sequences. Thus, by combining all the same-sequence reads together into a single read pack, which contains within it the read sequence and

the read count, we can greatly reduce the amount of work needed to extract barcode sequences and label barcode identifiers. For example, 438 million reads from the mRNA stability element library could be compressed into 101 million packs (a 4.3-fold reduction) (**Figure 6D**) while 81 million reads from the ribozyme library could be compressed into 11 million packs (a 7.4-fold reduction) (**Supplementary Figure 10**). Overall, the Oligopool Calculator carries out read filtering, merging, and packing with very high computational throughputs (181-270 million reads per hour with merging, 760 to 775 million reads per hour without merging) (**Table S3**). Several techniques are used to ensure that read processing is limited only by I/O throughput, even when using a typical multi-core desktop computer (**Methods**).

Rapid Counting of Packed Reads The final step of the Oligopool Calculator's analysis mode is to apply the Scry classifier on each barcode sequence within each read pack and compile a barcode count matrix that lists the number of times each reference barcode was found (**Figure 6E**). Two counting modes are available. When using association counting, each reference barcode sequence is uniquely associated with a sequence variant. The generated output lists the number of times each reference barcode sequence was observed (counts), including the barcode's unique identifier and the edit distance to the actual barcode sequence in the read. For example, the output will show the barcode counts for all perfectly matched barcode sequences (edit distances of 0) as well as the barcode counts of the imperfectly matched, but still uniquely identifiable, barcode sequences (positive edit distances). The barcode mapping process checks for the correctness of the anchor sequences and the overall amplicon architecture; if there are too many mismatches (above a researcher-defined threshold), then the read pack is discarded and not counted. When using combinatorial counting, the Scry classifier is applied to uniquely label each barcode sequence in the read pack and count the number of times each combination of reference barcodes are observed. Multiple barcodes can appear anywhere inside the read sequence, either in adjacent or distance locations, as defined by the researcher during the indexing step. The generated output lists the number of times each combination of reference barcode sequences was observed (counts). Missing barcodes are labeled as null values and treated as a potential combination, enabling their quantification for quality control.

In both counting modes, barcode sequences are extracted from reads by using the inputted anchor sequences and anchor-barcode distances to carry out a minimal, rapid alignment, followed by Scry classification and counting. At this step, the Oligopool Calculator can call a researcher-defined custom callback function, using the identified reference barcodes and read sequence as inputs, and tally the function's results in dictionary format. This optional callback function can be used to calculate another property of the genetic system variant that is found within the read pack, for example, identifying transcription start sites, RNA cleavage locations, or mutated amino acids. The callback function may also be used to apply additional quality control filters, such as discarding of concatemer reads. If the Scry classifier fails to find any expected barcode sequence, the Oligopool Calculator will automatically analyze the read sequence and determine if it contains PhiX genomic DNA or a low-complexity sequence. PhiX DNA is often added to deep amplicon libraries before sequencing, while low-complexity sequences are often generated if over-clustering or color saturation takes place during sequencing. The amounts of PhiX genomic DNA and low-complexity DNA are reported after the analysis is completed.

Overall, barcode mapping and counting is extremely fast after the read packs are generated. The Oligopool Calculator had a computational throughput of 1.15 to 1.2 billion reads per hour when analyzing the 590 million reads collected to characterize the promoter library (association counting)

whereas its throughput was 1.2 to 3.6 billion reads per hour when analyzing the 2 billion reads collected to characterize the mRNA stability elements and the 300 million reads collected to characterize the non-repetitive ribozymes (both using combinatorial counting) (**Table S4**). The count measurements are then used to quantify the activities of the genetic system variants; as an example, we measured the self-cleavage efficiencies of 6232 designed, non-repetitive ribozymes and identified over 300 with high self-cleavage efficiencies (**Figure 6F**).

The presented Oligopool Calculator is a suite of automated workflows that enables the rapid design of oligopools to carry out massively parallel construction of many genetic system variants as well as the rapid analysis of deep sequencing results to characterize the genetic system variants' activities. As illuminating examples, we show how the Oligopool Calculator was used to build and characterize thousands of promoter, mRNA stability, and ribozyme variants. Overall, the Oligopool Calculator now makes it much simpler to design complex oligopools with multiple well-designed barcodes, while eliminating common failure modes. After the experiments are finished and deep sequencing is carried out, the Oligopool Calculator converts what was an error-prone multi-step process into an hours-long automated procedure that generates precise measurements with transparent reporting. As oligopools are increasingly used to build and test thousands of genetic system variants, it will become ever more important to apply automated design and analysis workflows to extract re-usable knowledge from thousands of defined experiments.

Methods

Edge effect minimization: Given a non-empty set of n left/prefix context sequences X, and a non-empty set of n right/suffix context sequences Z, in the DNA alphabet set $\{\mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{C}\}$, we first extract all suffixes of all possible lengths from elements of X, and all prefixes of all possible lengths from elements of Z. Given a non-empty set of excluded DNA motif sequences E, we partition every element e_j in the set into all possible prefixes and suffixes of all possible lengths, where $0 \leq j < |E|$. A set of inserts $Y = \{y_1, y_2, \ldots, y_n\}$ are then designed to be free from left edge effects with respect to E, by ensuring that every prefix $y_i[0..q)$ is not a suffix of some excluded motif e_j, where $1 \leq q < |y_i|$, if placed next to x_i such that $x_i[p..|x_i|)$ is a prefix of e_j, where $0 \leq p < |x_i| - 1$, for all $0 \leq i < n$. Similarly, the insert y_i is optimized to be free from right edge effect with respect to an excluded motif sequence e_k, by ensuring that $y_i[u..|y_i|)$ is not a prefix of e_k, where $0 \leq u < |y_i| - 1$, when placed before context sequence z_i such that $z_i[0..v)$ is a suffix of e_k, where $1 \leq v < |z_i|$, for all $0 \leq k < |E|$. If the inserts are designed using Non-Repetitive Parts (NRP) Calculator in Maker Mode, these objectives are fulfilled using its traceback mechanism, otherwise the inserts are evaluated from a set of randomly generated sequence candidates until one meeting these criteria are found.

Barcode design: The barcode design algorithm requires as input barcode length integer b, oligo limit length integer o, minimum Hamming distance integer h, maximum shared repeat length integer k, the prefix context string set X, suffix context string set Z, and a set of excluded motifs E, and outputs barcode set C. All inputs are validated for data type and value range correctness. First, we verify (i) $4^b \geq n$, where n is the total number of variants in the oligopool, (ii) a barcode set can be built without any of the excluded motif sequence, i.e., there does not exist all possible 4^m strings of length m inside E, $\forall m > 0$, and (iii) adding a barcode of length b to the context X and Z doesn't exceed oligo limit o, i.e. $|x_{longest}| + |z_{longest}| + b \leq o$, where $x_{longest}$ and $z_{longest}$ are the longest sequences in X and Z. This is followed by construction of hash tables to store all information necessary to eliminate edge effects,

as well as building a k-mer dictionary from X and Z for eliminating repeats. We then initialize a random number generator to iteratively stream all numbers between 0 and $4^{b-1}-1$, and use it produce a set of random numbers $P=\{p_1,p_2,p_3,p_4\}$, where p_1 is the number produced by the generator, $p_2=4^b-1-p_1$, $p_3=\frac{4^b}{2}+p1$, and $p_4=\frac{4^b}{2}-1-p1$. Each of these numbers are then decoded as a base-4 integer and converted into a string in the DNA alphabet based on the mapping: $0\rightarrow\mathcal{A}$, $1\rightarrow\mathcal{C}$, $2\rightarrow\mathcal{G}$ and $3\rightarrow\mathcal{T}$. All elements in P are then evaluated for the following objectives.

Objective 1: $\mathrm{HammingDistance}(p_i,c)\geq h$, where $c\in C$ and $\mathrm{HammingDistance}(g,h)=\left|\left\{j:g_j\neq h_j,j=0,\dots,min(|g|,|h|)-1\right\}\right|$.

Objective 2: no edge effect occurs when p_i is accepted as c_j, where $j=|C|$, with respect to X and Z.

Objective 3: no k-mer in p_i is duplicated in context set X or Y upon acceptance as c_j.

Objective 4: no excluded motif $e\in E$ exists in p_i.

Objective 5: no substring of length q is shared between p_i and strings in C, where $q=\min(b,\lceil\log_4(n\times b)\rceil)$.

All barcode candidates that fulfill these objectives are sequentially added to C. To speed up objective 1, we use strategies described later for the Scry classifier to reduce the number of previously designed barcodes against which a current barcode is evaluated for Hamming distance conflicts. This process is repeated until $|C|=n$, or until we fail to produce a barcode in t successive attempts, where t is the number of trials in which a barcode is expected to be designed successfully. This parameter is initially set to 1000, but iteratively adjusted based on the failure count between successive barcode generation, by modelling the barcode design process as a Bernoulli trial.

Primer design: Primer design relies on NRP Calculator as an underlying component for sequence optimization. Input to the algorithm includes degenerate sequence constraint s in alphabet $IUPAC$ where each element in $IUPAC$ maps uniquely to $\mathbb{P}(DNA)$, oligo limit length integer o, primer type integer r, minimum and maximum melting temperatures t_{min} and t_{max}, maximum shared repeat length k, paired primer sequence s_p in DNA alphabet, a set of substrings from background DNA sequences B, prefix context string set X, suffix context string set Z, and a set of excluded motifs E. The primer sequence constraint s is verified to have degenerate $IUPAC$ characters, no palindrome substrings and no internal inverted repeats following canonical Watson-Crick base pairing, embedded within s. Oligo limit length and the excluded motif set is validated like those in barcode design algorithm. We then determine if a primer sequence can be designed within t_{min} and t_{max} bounds using Monte Carlo simulation of candidate primer sequences with predominantly weak or strong bases, i.e. $\mathrm{Tm}(s_{weak})\leq t_{max}$ and $\mathrm{Tm}(s_{strong})\geq t_{min}$, where s_{weak} and s_{strong} are candidate DNA strings preferentially composed using $\{\mathcal{A},\mathcal{T}\}$ and $\{\mathcal{G},\mathcal{C}\}$ respectively, and $\mathrm{Tm}(DNA^+)$ returns the melting temperature of a DNA string based on parameters from SantaLucia 1998 *(63)*. This is followed by construction of hash tables to store all information necessary to eliminate edge effects, as well as building a k-mer dictionary from X and Z for eliminating repeats. The primer sequence constraint s and the paired primer sequence s_p are then reverse complemented as necessary to correct orientation based on primer type parameter r. An instance of NRP Calculator is then used to design the primer sequence s', based on the following objectives with a jump count of $1{,}000$ and failure count of $100{,}000$.

Objective 1: no substrings are shared between s' and B.

Objective 2: no k-mer in s' is shared with context set X or Y.

Objective 3: no repeats longer than 6 characters in length are shared between s' and s_p.

Objective 4: no excluded motif $e \in E$ emerges in s' during sequence construction unless due to s.

Objective 5: no edge effect occurs at the terminal ends of s'.

Objective 6: post construction $t_{min} \leq \mathrm{Tm}(s') \leq t_{max}$.

Objective 7: no stable secondary structures forms between single stranded copies of s', i.e. $\mathrm{MFE}_{\mathrm{haiprin}}(s') \cong 0\ \mathrm{kcal} \cdot \mathrm{mol}^{-1}$, where $\mathrm{MFE}_{\mathrm{hairpin}}(DNA^{+})$ predicts the minimum free energy associated with the most favorable secondary structure for a given DNA string using Matthews 2004 parameters*(64)*.

Objective 8: no stable dimer forms between single stranded copies of s' and s_p, i.e. $\mathrm{MFE}_{\mathrm{dimer}}(s', s') \geq -7\ \mathrm{kcal} \cdot \mathrm{mol}^{-1}$ and $\mathrm{MFE}_{\mathrm{dimer}}(s', s_p) \geq -7\ \mathrm{kcal} \cdot \mathrm{mol}^{-1}$ also based on Mathews 2004 parameters.

All these objectives are fulfilled on a per nucleotide basis using a combination of local and global model functions input to NRP Calculator that is used to guide the primer sequence space exploration.

Spacer and motif design: Motifs are based on an $IUPAC$ sequence constraint s like a primer. Additional inputs include oligo limit length integer o, prefix and suffix context DNA string sets X and Z, and the excluded motif DNA string set E. If s has no degenerate characters, it is simply inserted as a constant between X and Z, and no optimization is done. Otherwise, NRP Calculator is used to design the motifs with respect to edge effect and excluded motif objectives like those specified for primer sequences. To speed up computation, once a candidate motif string is designed for context i it is validated against and assigned to as many remaining contexts as possible, such that $0 \leq i < n$ and $n = |X| = |Z|$. This process continues until every context has a designed motif. Spacers are a weaker form of motif, in which we derive a maximally degenerate $IUPAC$ sequence constraint s based either on a fixed or contextually defined spacer length integer l for every $x_i \in X$ and $z_i \in Z$ such that $|x_i| + |z_i| + l = o$. The objectives considered are same as motifs, but a spacer designed for context i is only evaluated for reassignment against context sequences x_j and z_j such that $|x_i| + |z_i| = |x_j| + |z_j|$.

Oligo splitting: The splitting algorithm takes as input a set of DNA strings $Y = \{y_1, y_2, \ldots, y_n\}$ where $n = |Y|$, split limit integer l, minimum melting temperature t, minimum Hamming distance integer h, and minimum and maximum overlap span length integers s_{min} and s_{max}. All inputs are validated for value range and correctness, and all $y \in Y$ are padded with random DNA strings to ensure $|y_i| = |y_j|$, for all $0 \leq i, j < |Y|$ and $i \neq j$. However, no splitting is done if the difference in the $|y_{longest}| - |y_{shortest}| > 0.5 \times l$. We first analyze Y to extract a set of tuples $C = \{\langle p_1, q_1 \rangle, \ldots\}$ such that $\mathrm{Entropy}(y_1[r..r+1) y_2[r..r+1) .. y_n[r..r+1)) \geq 0.25$ for all $p_k \leq r < q_k$, where $0 \leq k < |C|$, $\mathrm{Entropy}(y) = \sum_{x \in DNA} -\mathrm{p}(x) \times \log_4 \mathrm{p}(x)$, and $\mathrm{p(x)}$ is the probability of x in y. The strings in Y may then only be split in intervals from C. Based on l and C, we then choose a breakpoint w inside an interval tuple $c \in C$ such that $|y_i[u..w)| \leq l$, where u is set initially to 0. We then choose another breakpoint v also inside c, where $v < w$, such that all $z \in Z = \{y_1[v..w), y_2[v..w), \ldots, y_n[v..w)\}$ satisfy the following objectives.

Objective 1: $t_{min} \leq \mathrm{Tm}(z) \leq t_{max}$.

Objective 2: for all pairs of elements $z_i, z_j \in Z$, $\text{HammingDistance}(z_i, z_j) \geq h$, where HammingDistance function is same as the one evaluated for barcode design, $0 \leq i, j < |Z|$ and $i \neq j$.

Objective 3: $s_{min} \leq |z| < s_{max}$.

We initially set $v = w - o_{min}$ and then continue decreasing it by 1 until either all the objectives are met, or v reaches the starting coordinate in c. If no such breakpoint v is found, the algorithm terminates. Otherwise, the split fragment for all y is stored as $y[u..w)$, u is updated to v, and the remaining sections of the strings in Y are processed. The algorithm continues until the value of w is the ending coordinate of the last interval in C. The final set of split fragments are then $y[v..|y|)$. Once every string in Y has been split, the randomly added pads are stripped out from the final fragment set.

Padding of split sequences: A single set of split fragments Y are then padded with DNA strings until the length of each fragment reaches an oligo limit length integer o, with internal $3'$ flanking primers containing a TypeIIS restriction site of choice. The forward pad sequence constraint is maximally degenerate, and the full pad is based on the fragment length differences $\forall y \in Y$, a TypeIIS restriction site t, and the melting temperature range for the core primer which is kept approximately 20 bases long. Design of forward pad p is based on $d_{longest} = \left\lfloor \frac{|y_{longest}| - o}{2} \right\rfloor$ and $d_{shortest} = \left\lfloor \frac{|y_{shortest}| - o}{2} \right\rfloor$. Initially the algorithm specifies $p = t$. If $d_{shortest} < |t|$, then clearly no forward pad sequence can be designed to contain the cut site completely, and the algorithm terminates. Otherwise, p is prepended by $d_{longest} - |t|$ fully degenerate $IUPAC$ bases upstream such that $|p| = d_{longest}$, of which $p[|p| - \min(20, |t|)..|p|)$ is the primer region. The full pad sequence is then designed using NRP Calculator with previously defined primer objectives for the core primer region, and the only excluded motif set is $\{t\}$. A total of 34 restriction sites are available for use including AcuI, AlwI, BbsI, BccI, BceAI, BciVI, BcoDI, BmrI, BpuEI, BsaI, BseRI, BsmAI, BsmBI, BsmFI, BsmI, BspCNI, BspQI, BsrDI, BsrI, BtgZI, BtsCI, BtsI, BtsIMutI, EarI, EciI, Esp3I, FauI, HgaI, HphI, HpyAV, MlyI, MnlI, SapI, and SfaNI. These restriction enzymes are selected due to their short recognition sequence, and because they uniquely cut close to the $3'$ end of the recognition site. The reverse pad q is designed in the same way but within $o - \left|y_{longest}\right| - |p|$ bases, but in the reverse complement orientation. Once both pads are designed, the algorithm trims the $5'$ ends of p and q and appends them to $y \in Y$ in correct orientation such that $|y| = o$.

Scry classifier: The SCRY classifier essentially behaves like a 1-nearest neighbor model, since we are interested in locating the best reference barcode match given a sequenced barcode string. For training the model we instantiate a SCRY object which takes as input a set of DNA barcode strings X, a barcode label vector $Y \subseteq \mathbb{W}$ such that $|Y| = n$, barcode length integer b, and an expected sequencing error count integer t. We then store all inputs as object attributes and build the following data structures.

CONTIGS: an associative array where each key u_{Contig} is a substring of length l, where $b - t \leq l \leq b$, from some $x \in X$, and the value associated with u_{Contig} is the set of all $y_i \in Y$ such that u_{Contig} is a substring in x_i $\forall i \in [0..n)$.

SPECTRUM: an associative array where each key $u_{Spectrum}$ is a k-mer from X, and the associated value is a tuple ⟨IV, IG⟩, where $k = \lfloor b/2 \rfloor$. IV is the sorted set of all indexes i such that $x_i \in X$ contains $u_{Spectrum}$ as a k-mer inside it. IG is an array of length of $m = b - k + 1$ where the g^{th} element of IG is a tuple $\langle p, q \rangle$ such that every $x_h[g..g + k) = u_{Spectrum}$ $\forall h \in \text{IV}[p..q)$ and $\forall g \in [0..m)$.

For prediction tasks, the algorithm takes as input a sequenced barcode DNA string x' such that $b - t \leq |x'| \leq b + t$, otherwise it returns a tuple $\langle \emptyset, 0 \rangle$, where $\emptyset$ denotes an unmatched label. If we find an $x' = x_i \in X$, we simply return the tuple $\langle y_i, 1 \rangle$ as the corresponding result since it is a perfect match. Otherwise, we extract all substrings x^l of length l in decreasing order and check if $x^l \in$ CONTIGS, where $b - t \leq l < \min(|x'|, b)$. As soon as we find an x^l mapping uniquely to some $y_i \in Y$, we return the tuple $\langle y_i, l/b \rangle$ since it is a strong partial match. These steps are part of the fast prediction mode. Otherwise, we initialize an array of zeros SCORE of length n. Next, for each k-mer $u_{Sequence}$ from x' starting at location c, where $0 \leq c < |x'| - k + 1$, we use SPECTRUM to determine the set of indexes I_c such that $x_h[c' .. c' + k) = u_{Sequence}$ for some $c - t \leq c' \leq c + t$ and for all $h \in I_c$, and increase the count at those indexes in SCORE by k for each k-mer hit. Once all k-mers from x' have been processed, we determine all indexes I_{max} in SCORE containing the maximum value. For each such index $m \in I_{max}$, we then compute the $e = \text{EditDistance}(x_m, x')$, and accept y_m as the label for x' if $e \leq t$, where $\text{EditDistance}(DNA^+, DNA^+)$ returns the canonical Levenshtein distance *(65)* between two strings. If none of the indexes in I_{max} are accepted or more than one index is accepted, then the algorithm returns $\langle \emptyset, 0 \rangle$. Otherwise, it returns an unambiguous matching result $\langle y_m, \frac{b-e}{b} \rangle$. These steps constitute sensitive prediction mode. The algorithm runs fast prediction before executing sensitive matching steps.

Index construction: To build indexed objects of barcodes and associated variants, the algorithm requires a set of n tuples $B = \{\langle y_1, b_1 \rangle, \ldots\}$ where y_i is the identity label for DNA barcode string b_i and $0 \leq i < n$, barcode prefix and suffix constant strings $a^{barcode}$ and $c^{barcode}$, barcode prefix and suffix gap integers $g^{barcode}$ and $h^{barcode}$, and optionally a set of n tuples $V = \{\langle y_1, v_1 \rangle, \ldots\}$ where y_i is the identity label for DNA variant v_i, variant prefix and suffix constants $a^{variant}$ and $c^{variant}$, and variant prefix and suffix gap integers $g^{variant}$ and $h^{variant}$. Crucially, we interpret a barcode with label y to be linked with the variant which also has y as its label. All barcodes are required to be unique and fixed length, but the associated variants may have variable lengths or duplicates. If no associated variant information is provided, the indexed objects can only be used for combinatorial counting and not association counting. Once all inputs are validated, we build the following objects.

METAMAP: an associative array storing all meta data information such as variant count, barcode length, prefix and suffix constants, gap lengths, constant trimming policy and association information.

IDENTITYMAP: an ordered array of labels, i.e., IDENTITYMAP$[i..i+1) = y_i$.

BARCODEMODEL: a SCRY classifier trained on the barcode data and related information. The default error tolerance parameter $t_{default}$ is automatically inferred from barcode length and is set to $\left\lfloor \frac{|b_i|}{5} \right\rfloor$.

ASSOCIATEMAP: an ordered array of variant sequences, i.e., ASSOCIATEMAP$[i..i+1)$ contains variant with label y_i.

All objects are compressed, and stored in a single ".oligopool.index" file for later use. Collectively, an index serves as a single unit of information that needs to be mapped and counted in sequenced reads.

Read packing: The packing algorithm requires an R1 FastQ file r_1, R1 content type integer c_1, R1 Phred quality score threshold integer q_1, and R1 minimum read length integer l_1. If paired-end reads are available, additional inputs include R2 FastQ file r_2, R2 content type integer c_2, R2 quality threshold integer q_2, and R2 minimum read length integer l_2. Finally, the packing operation identifier integer p, batch size integer s, total number of CPU cores u to be used, and memory limit m to be allocated to

each core may be specified. All inputs are checked for correctness, and default values are used for optional inputs that are absent. Depending on p, if the paired-end reads are to be assembled, we process a sample of $50{,}000$ reads in r_1 and r_2 to determine the optimal assembly parameters. Briefly, we reverse complement both r_1 and r_2 as necessary based on c_1 and c_2, and align the $3'$ end of r_1 to $5'$ end of r_2 (innie alignment), as well as the $5'$ end of r_1 to the $3'$ end of r_2 (outie alignment) to rank the alignment scheme e with higher overlap length. The alignment overlap lengths are then stored in a vector O. We then define the read operation policy as the tuple $g = \langle a, o \rangle$ where a is a tuple $\langle e_{max}, e_{min} \rangle$ ranking both alignment schemes based on their frequency in the sampled reads, and $o = \max(\min(O), 10)$. If p indicates concatenation or if only r_1 is specified, then $g = \langle \oplus, \circ \rangle$ where $\oplus$ indicates the string concatenation operation and $\circ$ indicates the delimiter. Each of the u cores then process $\frac{1}{u}^{\text{th}}$ portion of r_1 and r_2 in parallel and stores the reads inside two process-specific associative arrays PACKHEADER and PACKLOCAL. Paired reads shorter than l_1 and l_2, or having quality scores less than q_1 and q_2 are filtered out and the read operation based on g is applied to produce a single read r_i for all $0 \leq i < |r_1|$. If merging is involved, we first apply scheme e_{max}, and then apply e_{min} if e_{max} produces an overlap of length $o_i < o$. Once the reads are aligned $r_i = r_{1,i}[0..|r_{1,i}| - o_i)r_{2,i}$. If $g_0 = \oplus$ then $r_i = r_{1,i} \circ r_{2,i}$. Next, we check if r_i exists as a key in PACKHEADER, which is initially empty. If r_i is a key in PACKHEADER, we simply increase its count value by 1, otherwise we store and update the count of r_i in PACKLOCAL until the cardinality of PACKLOCAL exceeds pack size s. Once the size of PACKLOCAL is greater than s and if PACKHEADER is empty, we move the contents of PACKLOCAL to PACKHEADER, otherwise we first serialize PACKLOCAL to disk and then clear its contents. Once all cores have finished processing r_1 and r_2, we move each read r from the j^{th} PACKHEADER to the 0^{th} PACKHEADER if r is a key inside the latter, and update their counts, where $1 \leq j \leq u$. Each updated PACKHEADER is then serialized to disk. Finally, all serialized content is compressed and archived in a single ".oligopool.pack" file. Each pack file is then a single experiment against which an index is to be mapped and counted. At any point during packing, if the memory occupied by the j^{th} process exceeds m after processing the i^{th} read, then the process is respawned where packing resumes at read number $i + u$. A process may also be restarted if the available free memory falls below 2 gigabytes. These policies allow packing to be executed in fixed memory and dynamically adjust the memory profile of each process.

Read counting: The combinatorial counting algorithm requires a single index file INDEXFILE, pack file PACKFILE, mapping type integer p, barcode error count integer $t_{barcode}$, associate error count integer $t_{associate}$, an optional callback function F, total number of CPU cores u to be used, and the memory cap per core m. In contrast, the association counting algorithm does not require associate error count as an input, but more than one index files ⟨INDEXFILE1, INDEXFILE2, …⟩ may be specified. Once all inputs are validated, read packs from the PACKFILE are distributed uniformly to each of the u cores for parallel mapping and counting. For association counting, each process loads all indexed objects from the input INDEXFILE and builds a set P containing 30-mers from the ΦX174 genome. We then process each PACK ∈ PACKFILE as an ordered array of tuples $\langle r, c \rangle$ where r is a DNA read string and c is the count of r in PACK. For each r_i, we first correct its orientation based on barcode prefix and suffix constants $a^{barcode}$ and $c^{barcode}$. If neither constants are matched in r_i, we check (1) if $|\{r_i[j..j+30) : r_i[j..j+30) \in P\}| > \frac{|r_i|-30+1}{2}$ where $j \in [0..|r_i| - 30 + 1)$, and (2) if $\mathrm{BaseCount}(r_i, d_k) = |r_i|$ where $\mathrm{BaseCount}(DNA^+, \Sigma)$ returns the sum of counts of each alphabet of Σ in a DNA string, $d = \mathbb{P}(DNA) \backslash \{\{\}, DNA\}$, and $0 \leq k < |d|$. If the first condition is met, we record r_i as a PhiX read. If the second condition is met, we consider r_i to be a low complexity read. If neither of the conditions are met, we simply record r_i as a "failed read".

Once the read has been corrected for orientation, we extract the region of r_i between the constants as barcode candidate string x_i' which is classified using the BARCODEMODEL stored in INDEXFILE based on p and $t_{barcode}$. If the barcode is not mapped, we discard r_i as a "failed read". Otherwise, we extract the region of r_i between the prefix and suffix constants $a^{variant}$ and $c^{variant}$ as the variant candidate string v_i', and check if $\mathrm{EditDistance}(v_i', v_i) < t_{associate}$, where v_i is the variant associated with predicted barcode label y_i. If v_i' shows higher edit distance, r_i is again considered as a "failed read". Next, if a callback function F is specified, r_i is discarded if $\mathrm{F}(r_i)$ returns $False$ (a "false read"), otherwise r_i is accepted. A central associative array COUNT which is accessible from all processes is then used to update the count of y_i by c. Once all packs are processed, the entries in COUNT are written out to a ".oligoopool.acount" file. Combinatorial counting follows the same steps as association counting, except associated variants cannot be mapped, but multiple barcodes may be mapped instead based on INDEXFILE1, INDEXFILE2, and so on. Each entry in the central COUNT is then a tuple $\langle y^1, y^2, \ldots \rangle$ where y^q is the barcode label predicted based on the q^{th} INDEXFILE. Counting processes are restarted in the same manner as done during read packing.

Code Availability

A Python implementation of Oligopool Calculator can be found at (1) github.com/hsalis/SalisLabCode, (2) github.com/ayaanhossain/oligopool and (3) pypi.org/project/oligopool.

Contributions

AH and HMS conceived the project, developed the algorithms, and wrote the manuscript. AH, TLL, DPC, JM, and HMS applied the algorithms and analyzed data. JM wrote documentation. All authors read and approved the final manuscript.

Supporting Information

Additional schematics, calculations, and computational benchmarks as shown in Supplementary Figures 1 to 10 and Supplementary Tables 1 to 4.

Acknowledgements

We'd like to thank Alexander Reis and David Kuo for their feedback. This project was supported by funds from the National Science Foundation (MCB-2131923), the Defense Advanced Research Projects Agency (FA8750-17-C-0254), and the Department of Energy (DE-SC0019090).

Potential Competing Interests

HMS is the founder of De Novo DNA. All other authors declare no competing interests.

References

1. Kosuri, S., and Church, G. M. (2014) Large-scale de novo DNA synthesis: technologies and applications, *Nature Methods 11*, 499.
2. Kosuri, S., Eroshenko, N., LeProust, E. M., Super, M., Way, J., Li, J. B., and Church, G. M. (2010) Scalable gene synthesis by selective amplification of DNA pools from high-fidelity microchips, *Nature Biotechnology 28*, 1295.
3. Palluk, S., Arlow, D. H., de Rond, T., Barthel, S., Kang, J. S., Bector, R., Baghdassarian, H. M., Truong, A. N., Kim, P. W., Singh, A. K., Hillson, N. J., and Keasling, J. D. (2018) De novo DNA synthesis using polymerase-nucleotide conjugates., *Nature biotechnology 36*, 645-650.

4. Kuiper, B. P., Prins, R. C., and Billerbeck, S. (2022) Oligo Pools as an Affordable Source of Synthetic DNA for Cost-Effective Library Construction in Protein- and Metabolic Pathway Engineering, *ChemBioChem 23*, e202100507.
5. Hossain, A., Lopez, E., Halper, S. M., Cetnar, D. P., Reis, A. C., Strickland, D., Klavins, E., and Salis, H. M. (2020) Automated design of thousands of nonrepetitive parts for engineering stable genetic systems., *Nature biotechnology 38*, 1466-1475.
6. Kosuri, S., Goodman, D. B., Cambray, G., Mutalik, V. K., Gao, Y., Arkin, A. P., Endy, D., and Church, G. M. (2013) Composability of regulatory sequences controlling transcription and translation in Escherichia coli, *Proceedings of the National Academy of Sciences 110*, 14024-14029.
7. LaFleur, T. L., Hossain, A., and Salis, H. M. (2022) Automated model-predictive design of synthetic promoters to control transcriptional profiles in bacteria, *Nature communications 13*, 5159.
8. Yokobayashi, Y. (2020) High-Throughput Analysis and Engineering of Ribozymes and Deoxyribozymes by Sequencing., *Accounts of chemical research 53*, 2903-2912.
9. Beltrán, J., Steiner, P. J., Bedewitz, M., Wei, S., Peterson, F. C., Li, Z., Hughes, B. E., Hartley, Z., Robertson, N. R., Medina-Cucurella, A. V., Baumer, Z. T., Leonard, A. C., Park, S.-Y., Volkman, B. F., Nusinow, D. A., Zhong, W., Wheeldon, I., Cutler, S. R., and Whitehead, T. A. (2022) Rapid biosensor development using plant hormone receptors as reprogrammable scaffolds, *Nature Biotechnology 40*, 1855-1861.
10. O'Connell, R. W., Rai, K., Piepergerdes, T. C., Wang, Y., Samra, K. D., Wilson, J. A., Lin, S., Zhang, T. H., Ramos, E. M., and Sun, A. (2023) Ultra-high throughput mapping of genetic design space, *BioRxiv* 2023.2003. 2016.532704.
11. Plesa, C., Sidore, A. M., Lubock, N. B., Zhang, D., and Kosuri, S. (2018) Multiplexed gene synthesis in emulsions for exploring protein functional landscapes, *Science 359*, 343-347.
12. Bryant, D. H., Bashir, A., Sinai, S., Jain, N. K., Ogden, P. J., Riley, P. F., Church, G. M., Colwell, L. J., and Kelsic, E. D. (2021) Deep diversification of an AAV capsid protein by machine learning., *Nature biotechnology 39*, 691-696.
13. Xiang, J. S., Kaplan, M., Dykstra, P., Hinks, M., McKeague, M., and Smolke, C. D. (2019) Massively parallel RNA device engineering in mammalian cells with RNA-Seq., *Nature communications 10*, 4327.
14. Klein, J. C., Agarwal, V., Inoue, F., Keith, A., Martin, B., Kircher, M., Ahituv, N., and Shendure, J. (2020) A systematic evaluation of the design and context dependencies of massively parallel reporter assays., *Nature methods 17*, 1083-1091.
15. Melnikov, A., Murugan, A., Zhang, X., Tesileanu, T., Wang, L., Rogov, P., Feizi, S., Gnirke, A., Callan, C. G., Kinney, J. B., Kellis, M., Lander, E. S., and Mikkelsen, T. S. (2012) Systematic dissection and optimization of inducible enhancers in human cells using a massively parallel reporter assay., *Nature biotechnology 30*, 271-277.
16. Movva, R., Greenside, P., Marinov, G. K., Nair, S., Shrikumar, A., and Kundaje, A. (2019) Deciphering regulatory DNA sequences and noncoding genetic variants using neural network models of massively parallel reporter assays., *PloS one 14*, e0218073.
17. White, M. A. (2015) Understanding how cis-regulatory function is encoded in DNA sequence using massively parallel reporter assays and designed sequences., *Genomics 106*, 165-170.
18. de Boer, C. G., Vaishnav, E. D., Sadeh, R., Abeyta, E. L., Friedman, N., and Regev, A. (2020) Deciphering eukaryotic gene-regulatory logic with 100 million random promoters., *Nature biotechnology 38*, 56-65.
19. Urtecho, G., Tripp, A. D., Insigne, K., Kim, H., and Kosuri, S. (2018) Systematic Dissection of Sequence Elements Controlling σ70 Promoters Using a Genomically-Encoded Multiplexed Reporter Assay in E. coli, *Biochemistry 58*, 1539-1551.
20. Cetnar, D. P., Hossain, A., Vezeau, G. E., and Salis, H. M. (2024) Predicting synthetic mRNA stability using massively parallel kinetic measurements, biophysical modeling, and machine learning, *Nature Communications accepted*,

21. Sample, P. J., Wang, B., Reid, D. W., Presnyak, V., McFadyen, I. J., Morris, D. R., and Seelig, G. (2019) Human 5' UTR design and variant effect prediction from a massively parallel translation assay., *Nature biotechnology 37*, 803-809.
22. Bogard, N., Linder, J., Rosenberg, A. B., and Seelig, G. (2019) A Deep Neural Network for Predicting and Engineering Alternative Polyadenylation., *Cell 178*, 91-106.
23. Najm, F. J., Strand, C., Donovan, K. F., Hegde, M., Sanson, K. R., Vaimberg, E. W., Sullender, M. E., Hartenian, E., Kalani, Z., Fusi, N., Listgarten, J., Younger, S. T., Bernstein, B. E., Root, D. E., and Doench, J. G. (2018) Orthologous CRISPR-Cas9 enzymes for combinatorial genetic screens., *Nature biotechnology 36*, 179-189.
24. Read, A., Gao, S., Batchelor, E., and Luo, J. (2017) Flexible CRISPR library construction using parallel oligonucleotide retrieval., *Nucleic acids research 45*, e101.
25. Shen, J. P., Zhao, D., Sasik, R., Luebeck, J., Birmingham, A., Bojorquez-Gomez, A., Licon, K., Klepper, K., Pekin, D., Beckett, A. N., Sanchez, K. S., Thomas, A., Kuo, C.-C. C., Du, D., Roguev, A., Lewis, N. E., Chang, A. N., Kreisberg, J. F., Krogan, N., Qi, L., Ideker, T., and Mali, P. (2017) Combinatorial CRISPR-Cas9 screens for de novo mapping of genetic interactions., *Nature methods 14*, 573-576.
26. Tomek, K. J., Volkel, K., Indermaur, E. W., Tuck, J. M., and Keung, A. J. (2021) Promiscuous molecules for smarter file operations in DNA-based data storage, *Nature Communications 12*, 3518.
27. Owczarzy, R., Tataurov, A. V., Wu, Y., Manthey, J. A., McQuisten, K. A., Almabrazi, H. G., Pedersen, K. F., Lin, Y., Garretson, J., McEntaggart, N. O., Sailor, C. A., Dawson, R. B., and Peek, A. S. (2008) IDT SciTools: a suite for analysis and design of nucleic acid oligomers., *Nucleic acids research 36*, W163-W169.
28. Untergasser, A., Cutcutache, I., Koressaar, T., Ye, J., Faircloth, B. C., Remm, M., and Rozen, S. G. (2012) Primer3--new capabilities and interfaces., *Nucleic acids research 40*, e115.
29. O'Halloran, D. M. (2016) PrimerMapper: high throughput primer design and graphical assembly for PCR and SNP detection., *Scientific reports 6*, 20631.
30. Tian, S., and Das, R. (2017) Primerize-2D: automated primer design for RNA multidimensional chemical mapping., *Bioinformatics (Oxford, England) 33*, 1405-1406.
31. Tian, S., Yesselman, J. D., Cordero, P., and Das, R. (2015) Primerize: automated primer assembly for transcribing non-coding RNA domains., *Nucleic acids research 43*, W522-526.
32. Hawkins, J. A., Jones, S. K., Finkelstein, I. J., and Press, W. H. (2018) Indel-correcting DNA barcodes for high-throughput sequencing., *Proceedings of the National Academy of Sciences of the United States of America 115*, E6217-E6226.
33. Buschmann, T. (2017) DNABarcodes: an R package for the systematic construction of DNA sample tags., *Bioinformatics (Oxford, England) 33*, 920-922.
34. Somervuo, P., Koskinen, P., Mei, P., Holm, L., Auvinen, P., and Paulin, L. (2018) BARCOSEL: a tool for selecting an optimal barcode set for high-throughput sequencing., *BMC bioinformatics 19*, 257.
35. Xu, Q., Schlabach, M. R., Hannon, G. J., and Elledge, S. J. (2009) Design of 240,000 orthogonal 25mer DNA barcode probes., *Proceedings of the National Academy of Sciences of the United States of America 106*, 2289-2294.
36. Estrada, J., Ruiz-Herrero, T., Scholes, C., Wunderlich, Z., and DePace, A. H. (2016) SiteOut: An Online Tool to Design Binding Site-Free DNA Sequences., *PloS one 11*, e0151740.
37. Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. (2016) Near-optimal probabilistic RNA-seq quantification., *Nature biotechnology 34*, 525-527.
38. Kim, D., Paggi, J. M., Park, C., Bennett, C., and Salzberg, S. L. (2019) Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype., *Nature biotechnology 37*, 907-915.
39. Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., and Kingsford, C. (2017) Salmon provides fast and bias-aware quantification of transcript expression., *Nature methods 14*, 417-419.

40. Gibson, D. G. (2009) Synthesis of DNA fragments in yeast by one-step assembly of overlapping oligonucleotides, *Nucleic Acids Research 37*, 6984-6990.
41. Gibson, D. G., Young, L., Chuang, R.-Y., Venter, J. C., Hutchison, C. A., and Smith, H. O. (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases, *Nature methods 6*, 343-345.
42. Daffern, N., Francino-Urdaniz, I. M., Baumer, Z. T., and Whitehead, T. A. (2024) Standardizing cassette-based deep mutagenesis by Golden Gate assembly, *Biotechnology and Bioengineering 121*, 281-290.
43. Roehner, N., Roberts, J., Lapets, A., Gould, D., Akavoor, V., Qin, L., Gordon, D. B., Voigt, C., and Densmore, D. (2024) GOLDBAR: A Framework for Combinatorial Biological Design, *ACS Synthetic Biology*
44. Espah Borujeni, A., Cetnar, D., Farasat, I., Smith, A., Lundgren, N., and Salis, H. M. (2017) Precise quantification of translation inhibition by mRNA structures that overlap with the ribosomal footprint in N-terminal coding sequences., *Nucleic acids research 45*, 5437-5448.
45. Espah Borujeni, A., Channarasappa, A. S., and Salis, H. M. (2014) Translation rate is controlled by coupled trade-offs between site accessibility, selective RNA unfolding and sliding at upstream standby sites., *Nucleic acids research 42*, 2646-2659.
46. Reis, A. C., and Salis, H. M. (2020) An Automated Model Test System for Systematic Development and Improvement of Gene Expression Models., *ACS synthetic biology 9*, 3145-3156.
47. Salis, H. M., Mirsky, E. A., and Voigt, C. A. (2009) Automated design of synthetic ribosome binding sites to control protein expression, *Nature Biotechnology 27*, 946-950.
48. Wong, A. S., Choi, G. C., Cui, C. H., Pregernig, G., Milani, P., Adam, M., Perli, S. D., Kazer, S. W., Gaillard, A., and Hermann, M. (2016) Multiplexed barcoded CRISPR-Cas9 screening enabled by CombiGEM, *Proceedings of the National Academy of Sciences 113*, 2544-2549.
49. Wong, A. S., Choi, G. C., Cheng, A. A., Purcell, O., and Lu, T. K. (2015) Massively parallel high-order combinatorial genetics in human cells, *Nature biotechnology 33*, 952-961.
50. Cheng, A. A., Ding, H., and Lu, T. K. (2014) Enhanced killing of antibiotic-resistant bacteria enabled by massively parallel combinatorial genetics, *Proceedings of the National Academy of Sciences 111*, 12462-12467.
51. Zhou, Y., Yuan, Y., Wu, Y., Li, L., Jameel, A., Xing, X.-H., and Zhang, C. (2022) Encoding genetic circuits with DNA barcodes paves the way for machine learning-assisted metabolite biosensor response curve profiling in yeast, *ACS Synthetic Biology 11*, 977-989.
52. Alcantar, M. A., English, M. A., Valeri, J. A., and Collins, J. J. (2024) A high-throughput synthetic biology approach for studying combinatorial chromatin-based transcriptional regulation, *Molecular Cell 84*, 2382-2396. e2389.
53. Hernandez Hernandez, D., Ding, L., Murao, A., Dahlin, L. R., Li, G., Arnolds, K. L., Amezola, M., Klein, A., Mitra, A., and Mecacci, S. (2023) Improved Combinatorial Assembly and Barcode Sequencing for Gene-Sized DNA Constructs, *ACS Synthetic Biology 12*, 2778-2782.
54. Stoler, N., and Nekrutenko, A. (2021) Sequencing error profiles of Illumina sequencing instruments, *NAR genomics and bioinformatics 3*, lqab019.
55. Sahlin, K., and Medvedev, P. (2021) Error correction enables use of Oxford Nanopore technology for reference-free transcriptome analysis, *Nature communications 12*, 2.
56. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020) Array programming with NumPy., *Nature 585*, 357-362.
57. Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011) ViennaRNA Package 2.0, *Algorithms for Molecular Biology 6*, 26.
58. Bolger, A. M., Lohse, M., and Usadel, B. (2014) Trimmomatic: a flexible trimmer for Illumina sequence data., *Bioinformatics (Oxford, England) 30*, 2114-2120.

59. Chen, S., Huang, T., Zhou, Y., Han, Y., Xu, M., and Gu, J. (2017) AfterQC: automatic filtering, trimming, error removing and quality control for fastq data., *BMC bioinformatics 18*, 80.
60. Magoč, T., and Salzberg, S. L. (2011) FLASH: fast length adjustment of short reads to improve genome assemblies., *Bioinformatics (Oxford, England) 27*, 2957-2963.
61. Zhang, J., Kobert, K., Flouri, T., and Stamatakis, A. (2014) PEAR: a fast and accurate Illumina Paired-End reAd mergeR., *Bioinformatics (Oxford, England) 30*, 614-620.
62. Gaspar, J. M. (2018) NGmerge: merging paired-end reads via novel empirically-derived models of sequencing errors., *BMC bioinformatics 19*, 536.
63. SantaLucia, J. (1998) A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics., *Proceedings of the National Academy of Sciences of the United States of America 95*, 1460-1465.
64. Mathews, D. H. (2004) Using an RNA secondary structure partition function to determine confidence in base pairs predicted by free energy minimization., *RNA (New York, N.Y.) 10*, 1178-1190.
65. Navarro, G. (2001) A guided tour to approximate string matching, *ACM Computing Surveys 33*, 31-88.