ACS SyntheticBiology

Letter

pubs.acs.org/synthbio

Merlin: Computer-Aided Oligonucleotide Design for Large Scale Genome Engineering with MAGE

Michael Quintin,*[,†,‡] Natalie J. Ma,[§,⊥] Samir Ahmed,[∥] Swapnil Bhatia,[∥] Aaron Lewis,[§] Farren J Isaacs,[§,⊥] and Douglas Densmore*[,∥]

[†]Program in Bioinformatics, Boston University, Boston, Massachusetts 02215, United States

[‡]Biological Design Center (BDC), Boston University, Boston, Massachusetts 02215, United States

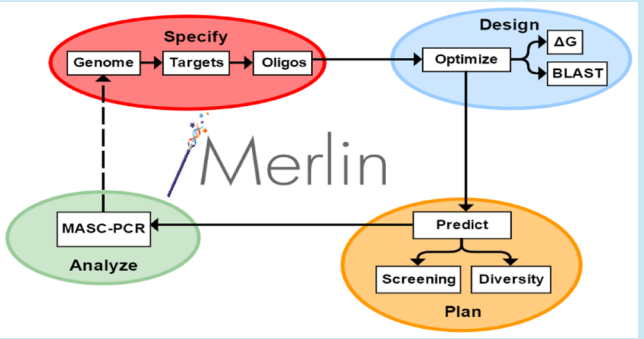
[§]Department of Molecular, Cellular, and Developmental Biology, Yale University, New Haven 06511, Connecticut, United States

[⊥]Systems Biology Institute, Yale University West Campus, West Haven, Connecticut 06516, United States

[∥]Department of Electrical and Computer Engineering, Boston University, Boston, Massachusetts 02215, United States

Supporting Information

ABSTRACT: Genome engineering technologies now enable precise manipulation of organism genotype, but can be limited in scalability by their design requirements. Here we describe Merlin (http://merlincad.org), an open-source web-based tool to assist biologists in designing experiments using multiplex automated genome engineering (MAGE). Merlin provides methods to generate pools of single-stranded DNA oligonucleotides (oligos) for MAGE experiments by performing free energy calculation and BLAST scoring on a sliding window spanning the targeted site. These oligos are designed not only to improve recombination efficiency, but also to minimize off-target interactions. The application further assists experiment planning by reporting predicted allelic replacement rates after multiple MAGE cycles, and enables rapid result validation by generating primer sequences for multiplexed allele-specific colony PCR. Here we describe the Merlin oligo and primer design procedures and validate their functionality compared to OptMAGE by eliminating seven AvrII restriction sites from the *Escherichia coli* genome.



KEYWORDS: *bioengineering, genomics, multiplex automated genome engineering, MAGE, synthetic biology, automation, software*

Multiplex automated genome engineering (MAGE)[1] is a genome editing technique that utilizes homologous recombination proteins originally isolated from phage λ to achieve scarless incorporation of synthetic ssDNA oligonucleotides (oligos) into a bacterial genome.[2−5] These oligos consist of 90 nucleotides, with mutations in the central region flanked by 5′ and 3′ homology to the target site. Once introduced into the cell by electroporation, oligos bind the λ-beta protein. Data suggests these oligos then anneal to the lagging strand of the replicating bacterial chromosome and introduce the mutation.[6] Subsequent rounds of replication stably fix the mutation. MAGE has been used to construct highly modified organisms,[7−10] inducing genomes with multiple short (1−60 bp) specific sequence changes (*i.e.*, mismatch, deletion or insertion) at many targeted loci (>300). Alternatively, a pool of degenerate oligonucleotide sequences can be used to create diverse populations and explore a large genotypic landscape. However, genome modification efforts that aim to accumulate many (>100) mutations in a single genome require extensive time to design and multiple successive MAGE cycles and would benefit from robust automated solutions.

Computer-aided design software plays an important role in synthetic biology.[11] Many seemingly straightforward tasks such as designing PCR primers or creating MAGE oligos with the modified bases at a predetermined position can be performed "well enough" by hand, yielding results with suboptimal designs and saving effort at the cost of efficiency and, ultimately, turnaround time. Automated processes allow design decisions to be reproduced, archived, standardized, and shared, easing the integration of new techniques or insights as they become available. By incorporating tools into the experiment design pipeline such as thermodynamic modeling, off-target interaction prediction, and result simulation, we can provide a framework to help guide the design process. Merlin is built with the intention of increasing the rigor applied to oligo design in order to improve performance without requiring extra effort on the user's part.

Merlin provides an integrated software environment for the design of MAGE experiments which replaces manual oligo design processes, provides additional validation utilities, and is open source and extensible to allow for community development (Figure 1).

Special Issue: IWBDA 2015

Received: November 3, 2015

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX

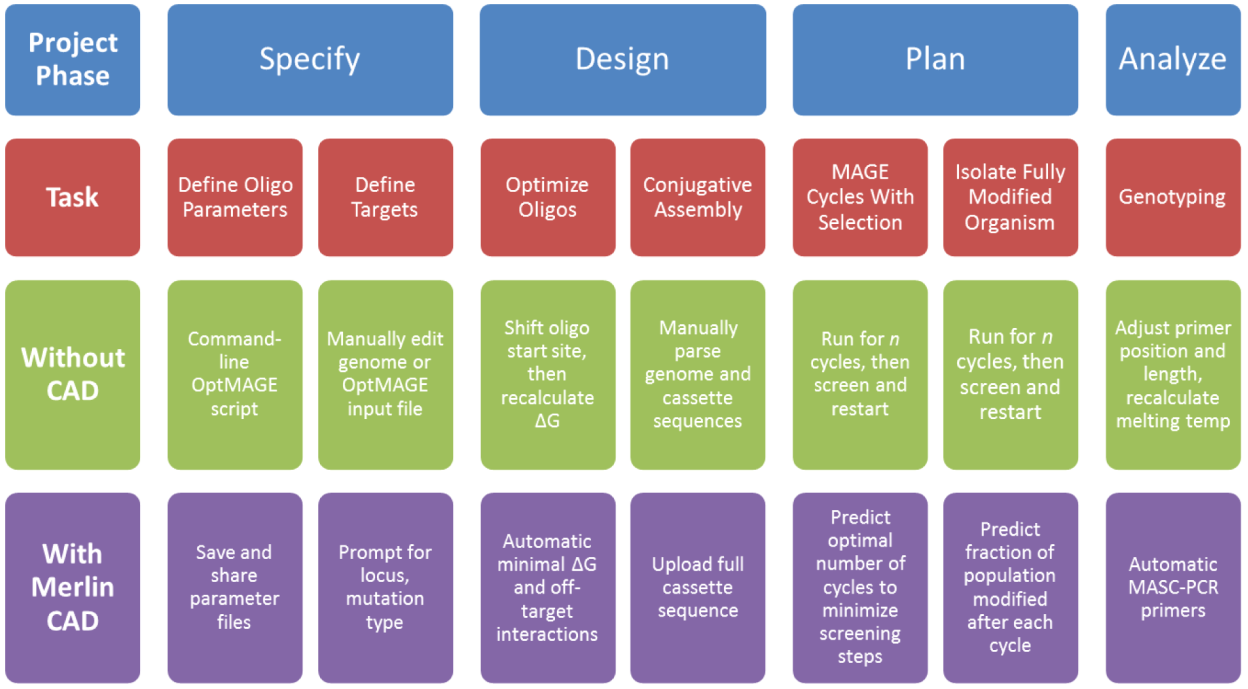


Figure 1. Illustration of how Merlin assists in all phases of the biodesign automation framework.[11] Specific tasks reflect changes in our own workflow due to computer-aided design. Notable modifications to the Specify phase include removing the need to manually edit parameter files and interact with the command-line OptMAGE script. The Design phase is improved by removing the need to edit sequences by hand. Oligo efficiency prediction informs how many MAGE cycles will be run between screening steps. Finally, automatic generation of MASC-PCR primers removes the need for tedious trial-and-error to assemble a set of primers with an acceptable range of melting temperatures.

While Merlin is not the first software tool for MAGE oligo design, the approach we have taken is to fulfill the needs of an experimentalist as opposed to a bioinformatician by removing typical obstacles to the adoption of CAD software (*e.g.*, programming language requirements, monetary cost, and limited or specific functionality). The user interface is designed to guide users through the considerations in the process of creating oligos by presenting recommended parameters that the user may modify. Modified parameter files can be exported to preserve a record of the settings used, and uploaded in order to recreate experiments with the same parameters. Each optimized oligo is presented with details summarizing why its specific location on the target genome was selected by showing the folding energy of every possible oligo spanning the same locus as well as the relative likelihood of off-target interactions as determined by BLAST. A model for the expected allelic replacement rate after each MAGE cycle is included to allow the user to estimate how many cycles are required to create a population of the desired diversity, or to calculate how many colonies should be screened to find a sample with the desired number of alterations. To support projects in which distinct genomic regions are modified by MAGE and will be combined by conjugative assembly genome engineering (CAGE), Merlin is capable of generating primer sequences for dsDNA cassettes to be used in the CAGE procedure.[12] Finally, Merlin can create sets of MASC-PCR primers to enable rapid screening of the modified genomes that are produced.

MAGE experiments typically consist of multiple cycles of transformation to introduce the mutagenic oligo pool into the cell. Merlin is capable of creating visualizations based on the calculated probability of each oligo becoming incorporated in the target genome for each cycle (Figure 2). These statistics are useful for predicting how many cycles will be necessary to create a population with a specific diversity of modifications, or how many cycles will be needed to produce an organism that is modified at all target sites.

Multiplexed allele-specific colony PCR allows for simultaneous screening of short mutations at many nonoverlapping loci in a single PCR reaction by generating DNA fragments of different sizes for each locus.[7] The size of these fragments is set such that each can be easily distinguished, producing distinct genotypic "barcodes" on a gel. This technique is valuable for reducing the manual labor and expenses involved in interrogating multiple genome modification events. Generating a set of primers for wild-type and mutant genotypes of each target can be tedious, particularly when screening for short sequence changes where the melting temperature of all primers should ideally be within 1.5 °C. Melting temperatures can be adjusted by shifting the location of a primer or by changing its length, but doing so manually may require hours of trial and error. Merlin is capable of generating these primer sets automatically, with PCR product lengths defined by the user. For discussion of the algorithm used to create primers, refer to the Methods.

A well-described approach[7,12] to large-scale genome engineering, conjugative assembly genome engineering (CAGE), involves targeting distinct genomic regions in parallel experiments then assembling a chimeric genome from multiple mutant strains. Because MAGE and CAGE are closely associated, Merlin has been built with the capability to generate sequences for the sets of primers for dsDNA cassettes that are necessary for the recombineering of selectable markers used in CAGE. The user is prompted to provide the DNA sequence of the selectable marker they wish to integrate into the donor genome and the start and end positions of the desired replacement on the currently loaded recipient genome. Merlin then generates 60 bp primer sequences for the selectable marker cassette with overhangs for integration into the appropriate site in the donor genome, enabling integration of the

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX

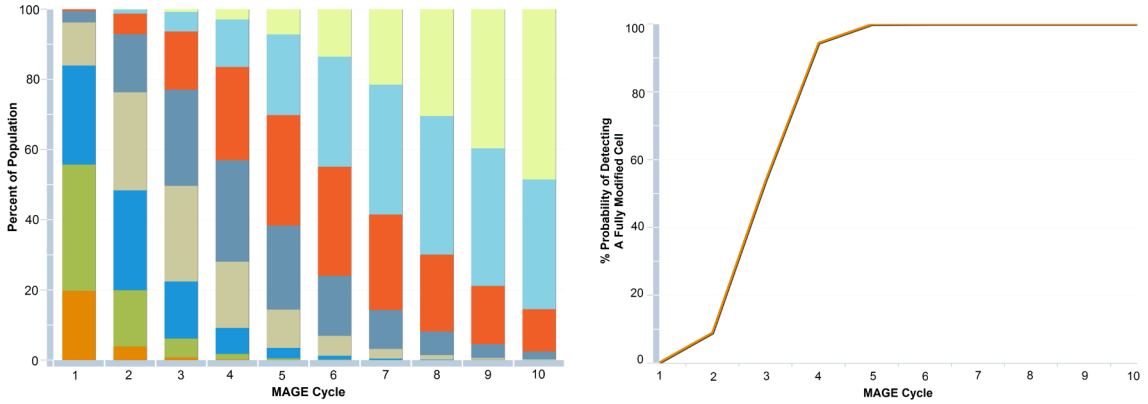


Figure 2. Graphical output as generated by Merlin showing results for oligo replacement rate calculations displayed in two different formats for the same experiment. Left: Genotypic diversity expressed as percent of the bacterial population expected to have 0 through *n* modifications after each MAGE cycle. A mouseover tooltip (not shown) displays the calculated percentage and the number of modifications for each segment. $n = 7$, segments are stacked in ascending order with orange depicting 0 modifications, green depicting 1, *et cetera*. Right: Predicted likelihood of detecting a bacterium that has incorporated all oligos after each cycle if 96 colonies are screened.

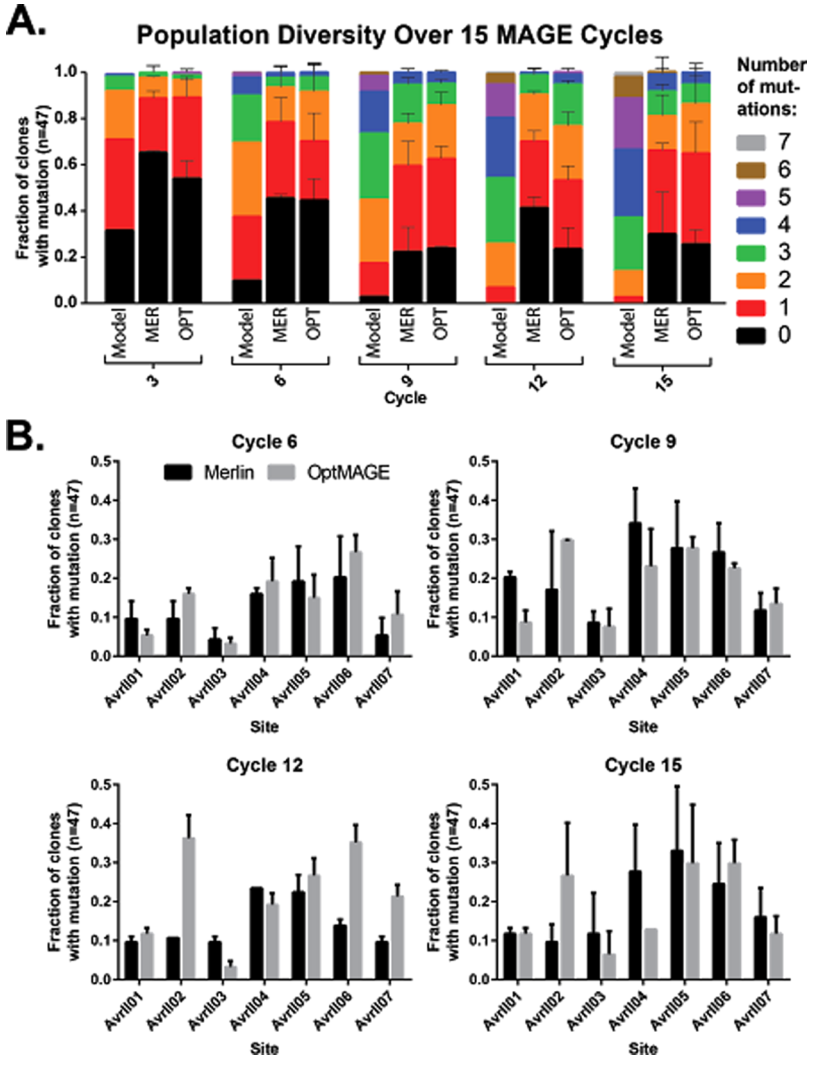


Figure 3. Oligos designed *via* Merlin and OptMAGE generate similar mutation diversity. (A) Population diversity over 15 MAGE cycles as expected by Merlin's ARE prediction model (left) and using oligos generated by Merlin (center) and OptMAGE (right). (B) Allelic replacement rates for each oligo show greater variation between loci than would be expected from the calculated ΔG. This suggests that specific structural features or genomic context may play a role in determining the likelihood of recombination. Allelic replacement rates are measured using validated MASC-PCR primers generated by Merlin. Comparable results were observed after 3 cycles (Figure S2). Error bars are the range of values for two separate populations. $n = 47$.

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX

marker into the donor genome and subsequent merging of donor and recipient genome *via* CAGE.[7,12]

To test Merlin's design capabilities, we used MAGE oligos designed by both the Merlin algorithm and Merlin's integrated version of the OptMAGE script[1] and PCR primers generated by Merlin to induce and assay for modifications in a population over multiple MAGE cycles. We created oligos targeting seven of the 16 AvrII restriction sites, then used these to perform 15 rounds of MAGE in the *mutS* knockout *E. coli* strain EcNR2.[1] We first validated the MASC-PCR primers from the Merlin algorithm by performing three cycles of MAGE with Merlin oligos and then assaying forty-eight clones with MASC-PCR primers and Sanger sequencing (Figure S1). Results were identical for each data set, demonstrating the accuracy of the MASC-PCR primers (Table S1). We then assayed the diversity in Merlin and OptMAGE populations at cycles three, six, nine, 12, and 15 in two duplicate populations using MASC-PCR. Overall, we found that both Merlin and OptMAGE generate similar population diversity over multiple cycles, indicating that overall efficiency was similar for oligos designed from both Merlin and OptMAGE and validating the Merlin oligo design algorithm (Figure 3A).

In each case, a significant portion of the population did not acquire mutations at any of the targeted loci. Several explanations for this phenomenon are possible, making further study into the MAGE process necessary before it can be incorporated into the model. MAGE modification may not be fitness-neutral as we assume when creating intergenic or silent mutations, leading to slower replication in mutants relative to the unmodified population. Alternatively, there may be a distinct subpopulation of mutant cells with reduced competency for oligo uptake. It should be noted that Pósfai *et al.* observed increased competency for DNA uptake in *E. coli* strains modified to lack insertion sequence elements and prophages in their genomes, suggesting a genetic component to competency.[13]

To further compare the Merlin and OptMAGE design algorithms, we examined the efficiency of specific oligos. For most sites, rates of mutagenesis were similar for both Merlin and OptMAGE oligos. However, we found that the OptMAGE oligo demonstrated greater efficiency than the Merlin oligo at target site two (AvrII02) (Figure 3). The AvrII02 OptMAGE oligo (AvrII02-Opt) generated mutations in up to 35% of the cell population while the Merlin oligo (AvrII-Mer) generated mutations in up to 18% of the cell population (Figure 3B). This difference cannot be explained by the free energy value of the oligos, as the Gibbs free energy of the OptMAGE oligo was calculated to be −8.08 kcal/mol and the free energy of the Merlin oligo was −11.30 kcal/mol; both are well within the acceptable range for MAGE oligos. We examined the structure of both oligos using the RNAFold program from the ViennaRNA package with the included Mathews model for DNA energy parameters[14] and found two potential factors that may explain this result: (i) A 5′ stem loop in AvrII02-Mer, and (ii) A 3′ stem loop in AvrII02-Opt (Figure S4). Both loops are present in each oligo, but their relative positions to the ends of the oligos are noteworthy; either may have an impact on ARE. Because of the implication that free energy scores do not reflect the full impact of oligo folding on MAGE efficiency, these factors may inform future improvements to the Merlin optimization algorithm.

METHODS

Oligo Design. The Merlin algorithm for generating an oligo pool from a list of target locations on the genome is as follows. Required inputs for each individual oligo are the start and end (inclusive) positions x_{start} and x_{end} of the modification on the genome, the *replichore* (1 or 2) and *strand* (+ or −) on which the target is located, the mutation *type* (insertion, deletion, or mismatch), and the sequence s' for a mismatch or insertion mutation. Global parameters for all oligos are the genome sequence S, the oligo length l, the minimum number of unmodified nucleotides that buffer the ends of an oligo $b_{3'}$ and $b_{5'}$, and the Gibbs free energy threshold G_t above which secondary structures in the oligonucleotides are not expected to interfere with binding. Subsequences will be denoted $S[start,end]$with both positions inclusive.

ΔG is calculated by RNAFold in the ViennaRNA package with the arguments *"-P dna_mathews.par−noconv−noPS"*.[14] *bitscore* and *evalue* are calculated from blastn with arguments *"-word_size 11 -evalue 10"*.

1. If *type* is *insertion* or *mismatch*, orient the target so the oligo will match the sequence of the lagging strand during replication.
   1.a. If $\{(replichore = 2)\ \&\ (strand = "-")\}$ or $\{(replichore = 1)\ \&\ (strand = "+")\}$: $s' = ReverseComp(s')$
2. Isolate the window of nucleotides that can be considered for inclusion in the oligo.
   2.a. $s_{pre} = S[x_{start} - 1 - l + |s'| + b_{3'}, x_{start} - 1]$
   2.b. $s_{post} = S[x_{end} + 1, x_{end} + 1 + l - |s'| - b_{5'}]$
3. Reorient the pre- and postsequence to place them on the lagging strand.
   3.a. If $(replichore = 1)$:
      3.a.i. $s_{pre} = ReverseComp(s_{pre} + s_{post})\ [1,|s_{post}|]$
      3.a.ii. $s_{post} = ReverseComp(s_{pre} + s_{post})\ [|s_{post}| + 1, |s_{pre} + s_{post}|]$
4. View a sliding window across the assembled sequence and assign scores.
   4.a. Initialize two vectors of length l: the free energy score E and the BLAST result score against the input genome B for each position.
   4.b. For all $s = (s_{pre} + s' + s_{post})\ [i,i + l]$ where $i \in 1 \cdots |s_{pre}|$:
      4.b.i. If $G(s) > G_t$: $E[i] = 0$
      4.b.ii. Else $E[i] = -G(s)$
   4.c. $B[i] = bitscore(s) * e^{-e\text{-value}(s)}$
5. Select an optimized oligo by first selecting the oligos with the lowest free energy score, then choosing any oligo from that set that minimizes the BLAST score.
   5.a. $\vec{j} \subset \{1 \cdots l\} s.t.\ \forall j: E[j] = min(E)$
   5.b. $k \in \vec{j}\ s.t.\ B[k] = min(B[\vec{j}])$
   5.c. $s_{opt} = (s_{pre} + s' + s_{post})\ [k,k + l]$

This method is an improvement on OptMAGE by virtue of calculating values for all possible oligos, as well as by the addition of the BLAST step. The Merlin interface graphically reports the free energy and sequence homology results for every potential position, highlighting the difference between each Merlin oligo and its OptMAGE-produced counterpart targeting the same locus (Figure S3).

Oligo Efficiency Prediction. The allelic replacement efficiency (ARE) over multiple cycles can be modeled as a Poisson binomial distribution in cases where modifications are occurring at discrete sites.[15] The average ARE is determined, then modified by an empirically determined "pooling factor" that accounts for decreased efficiency proportional to increasing complexity in the oligo pool.

The prediction function for individual oligo AREs is based on fitting empirical data from previous studies[1,15] with parameters

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX

that vary by the modification type. The equation used is of the form

$$\hat{\mathrm{ARE}} = \mathrm{RE}_0 * e^{\alpha(l-1)}$$

where l is the length of the insertion or deletion or the number of of modified bases in a mismatch mutation, and RE_0 and α correspond to the modification type as in Table 1.

Table 1

	RE_0	α
mismatch	0.26	−0.135
insertion	0.15	−0.075
deletion	0.23	−0.058

As the number of distinct oligo sequences included in the pool increases, the ARE of each individual oligo decreases. This "pooling factor" accounts for oligo-to-oligo interactions and the relative dilution of each sequence in a fixed number of oligos by reducing each ARE with the formula[15]

$$\hat{\mathrm{ARE}}_p = \hat{\mathrm{ARE}} * (1 - e^{-1.59/n})$$

where n is the number of oligos in the pool. Using the ARE as the percent likelihood of an oligo being incorporated in a single MAGE cycle, and on the assumption that each consecutive cycle can be considered an independent trial, the probability of a given oligo being incorporated after c cycles is

$$P = 1 - (1 - \hat{\mathrm{ARE}}_p)^c$$

The above can now be used to calculate the expected distribution of oligo integration after a number of cycles as a percentage of the total cell population. The full equation for the probability mass function of the Poisson binomial distribution is given as

$$P(K = k) = \Sigma_{A \in F_k} \Pi_{i \in A} P_i \Pi_{j \notin A} (1 - P_j)$$

This equation may be used to determine the percentage of the population expected to have k modifications, where F_k is the set of all sets of size k of nonoverlapping oligos. It should be noted that in practice the size of F_k grows quickly with increasing n, so Merlin instead incorporates a method to iteratively combine the individual predicted AREs of each oligo after each cycle (see the next section).

Population Diversity Prediction from AREs. The algorithm Merlin uses to construct a data table showing the fraction of the population expected to incorporate k out of n oligos through C cycles is as follows:

1. Initialize an $n + 1$ by C table T
2. Set the last column $T(1,C) = 1$, all other values = 0
3. For $c \in \{1,...,C\}$, $k \in \{2,...,n+1\}$, $j \in \{k,...,2\}$:
   - 3.a. Calculate the pooled ARE for the kth oligo as described above and find the probability p of incorporation after c MAGE cycles
     - 3.a.i. $\hat{\mathrm{ARE}}_p = (\mathrm{RE}_0 * e^{\alpha\,(l-1)}) * (1 - e^{-1.59/n})$
     - 3.a.ii. $p = 1 - (1 - \hat{\mathrm{ARE}}_p)^c$
   - 3.b. Set $T(j,c) = T(j,c) + T(j-1,c) * p$
   - 3.c. Set $T(i,c) = T(i,c) * (1 - p)$ where $1 \le i < j$

MASC-PCR Primer Optimization. In a multiplexed PCR reaction, it is important that the melting temperature of each primer is the same. In general, all temperatures should be within a 1.5 °C range as further deviation will make single base pair changes difficult to detect.

Merlin predicts melting temperatures using the BioPython Bio.SeqUtils.MeltingTemp module.[16] The user is prompted to provide ion and nucleotide concentrations matching their preferred PCR protocol, as well as to define how much variability should be allowed in primer size and the length of the resulting DNA fragments.

Instead of discussing the "upstream" and "downstream" PCR primers, it is convenient to consider the "targeted region" and "nontargeted region" primers instead. The primers in the nontargeted region fall outside of the sequence covered by the MAGE oligos since typical oligo size is shorter than the smallest recommended amplicon size.

The melting temperature of a PCR primer can be changed by varying its length or repositioning the targeted region to alter the primer sequence. In the case of targeted region primers the 3′ position is fixed, so melting temperature can only be controlled by changing the length from the 5′ end.

For a primer with a length that may vary by size l and a start position that may vary by size s, there are $l*s$ possible sequences to consider. In the default case, a 100 ± 15% bp amplicon with a primer length of 16−30, this gives 450 sequences. To increase the speed of the optimization process, a simulated annealing technique is applied as follows. We will refer to the "temperature" of the algorithm as T, to avoid confusion with the target melting temperature for the PCR reaction.

1. Construct an $l*s$ empty matrix M.
2. Draw primers at random to calculate an initial T_0.
   - 2.a. A number of random uphill transitions are made.
     - 2.a.i. Randomly select two adjacent coordinates on M.
     - 2.a.ii. Calculate the melting temperature of the primer sequence with the corresponding length and start position.
     - 2.a.iii. The objective function score is based on the target melting temperature t_m, and defined as $f(t_1,t_2) = |t_m - t_1| - |t_m - t_2|$. A positive score indicates the second primer is closer to having the desired melting temperature.
   - 2.b. The average absolute value of the objective function score, δ, is calculated.
   - 2.c. The initial temperature is calculated with the equation $T_0 = -\delta/(\ln P)$ where P is the initial acceptance probability, the fraction of uphill transitions that are accepted. P begins low (50%) and increases with time.
3. Begin at an initial coordinate at the center of M.
4. Iterate until an acceptable result or inescapable local maximum is found:
   - 4.a. Select another coordinate by a random walk of *round* $[(w * (T_0 - T)/(T_0 - T_{\mathrm{final}})]$ steps from the selected point, where w is some initial step size that begins large (*e.g.*, the length of the longest side of M) and will diminish with time.
   - 4.b. Calculate the melting temperature of the primers corresponding to each coordinate and store them in M.
   - 4.c. Calculate the score $f(t_1,t_2) = |t_m - t_1| - |t_m - t_2|$.
   - 4.d. If $f > 0$, transition to the new coordinates with probability $e^{-f/T}$.
   - 4.e. If the melting temperature of the selected primer is within some threshold of the final target temperature, return that primer and terminate.
   - 4.f. After some number of iterations, reduce T. This problem space is small enough that a simple linear decrease is sufficient, such as $T' = T - xT_0$ where $x < 1$.

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX

4.g. If $T = T_{final}$, locate the coordinate with the closest melting temperature to the target, return the corresponding primer and terminate.

MASC-PCR Primer Pool Generation. Primers in the targeted region are under tighter constraints than in the nontargeted region, so targeted region primers are created first by the program. These should start at specific locations, placing the first modified base of the mutated DNA sequence (or the corresponding unmodified base) at the 3′ position to maximize specificity. To preserve clarity of the barcode-type readout and enable both species to use the same nontargeted region primer, the wild type and mutant primers should have the same orientation. The methodology Merlin uses to create the primer pool is as follows.

Create targeted region primers:

1. For each oligo, four primer sequences will be generated. These match mutant and wild type pairs of loci, in each orientation.
2. The location of the 3′-most base on the primer depends on the targeted mutation type. For missense mutations, the location of the first (or last, depending on orientation) modified base is used. For insertions, the first inserted base is used. For deletions, the primer is positioned so that the deleted region is 1/4 of the primer length away from the 3′ end. Each mutant/wild type pair of primers uses the same genomic coordinate for the 3′ base.
3. Optimize each of the primers as described in the previous section, with the start position fixed.
4. If no pair of optimized primers has a melting temperature outside of the acceptable range, select the pair that has the lowest total difference from the target melting temperature. If only one pair has a primer that does not fit within the range, select the other pair. If both pairs are outside the acceptable range, select the pair with the lowest maximum difference from the target.

Create nontargeted region primers:

1. For the pair of targeted region primers selected at each locus, and for each amplicon size specified by the user, create an optimized primer either up or downstream of the targeted region based on the orientation of the selected pair.
2. Construct m lists of length equal to the number of desired amplicon sizes, such that $m * n$ amplicons > n oligos.
3. For each set of primers corresponding to a single target, if no primer is within the acceptable threshold of the target melting temperature, select the primer that is closest and store it in an empty spot corresponding to its length on some list. If no lists have a vacancy, displace a primer that remained inside the threshold. If this is not possible, create a new list to occupy.
4. Fill the remaining empty slots in each list by searching the primers whose target is not yet accounted for and that correspond to the slot's amplicon size, and choose the primer with the closest melting temperature to the target. To make the results easier to read, prioritize amplicon lengths by skipping every second length at first to maximize the space between bands.

Validation Methods. For validation, we used strain EcNR2[1] and MAGE was performed as described previously.[6] To identify target loci randomly spaced around the genome, we analyzed the *E. coli* MG1655 genome for AvrII restriction sites and identified 16. We chose seven of these AvrII restriction sites to target because they occurred in noncoding regions or in genes where silent mutations were possible, minimizing the chance of fitness defects that would bias calculation of mutation efficiencies. Every 3 cycles, 50 μL of MAGEd culture diluted 1:10 000 was plated on LB Lennox agar plates containing chloramphenicol or carbenicillin (markers carried by ECNR2) to prevent contamination. After growth overnight, 47 colonies were picked into a 96-well plate, with two wells containing ancestral strain to benchmark MASC-PCR primers on.

For genotyping with MASC-PCR, we used methods described previously.[7,12] PCR reactions were run according to Kapa Biosystems Multiplex Mix specifications, with an annealing temperature of 64.5 °C and extension time of 1 min for 28 cycles.

Infrastructure. The interface for Merlin is built using VectorEditor (available at https://github.com/JBEI/vectoreditor/), an open source web based DNA sequence analysis and editing tool maintained by the Joint BioEnergy Institute (JBEI).[17] Melting temperature calculations are obtained from the Bio.SeqUtils.MeltingTemp module in BioPython[16] (version 1.65 at the time of this submission). Free energy calculations are performed with the RNAFold program (version 2.1.9 at the time of this submission) from the ViennaRNA package from the Theoretical Biochemistry Group within the Institute for Theoretical Chemistry at the University of Vienna, using the included Mathews model for DNA energy parameters.[14]

Java, Python, and ActionScript source code for Merlin is available at the CIDAR Github repository (https://github.com/CIDARLAB/) under the BSD 3-Clause License (http://opensource.org/licenses/BSD-3-Clause).

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acssynbio.5b00219.

MASC-PCR gel used for primer validation (Figure S1), Sanger sequencing and PCR validation of Merlin oligo efficiency (Table S1), Comparison of Merlin and OptMAGE oligos after 3 MAGE cycles (Figure S2), Graphical output of oligo optimization (Figure S3), Structural comparison of AVRII02 oligos (Figure S4), Oligo sequences and predicted structures (Table S2). (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

*E-mail: mquintin@bu.edu.

*E-mail: dougd@bu.edu.

Author Contributions

D.D. and S.B. conceived and supervised the project, and designed the implementation. S.A. wrote the first version of Merlin, which was revised and extended by M.Q., who also wrote the manuscript. N.J.M., A.L., and F.J.I. designed or executed the experimental validation and provided feedback on feature requirements and the software implementation. All authors read and edited the manuscript.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors thank Harris Wang for providing early access to and guidance in using Optmage, and Harris Wang and George Church for valuable brainstorming discussions during the early

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX

stages of this work. We also thank A. Berliner and J. Hodgson for early discussions relating to this project. M.Q was funded by the National Science Foundation under award #1253856. N.J.M. gratefully acknowledges support from the National Institutes of Health (T32GM007499, T32GM007223) and the Gruber Foundation.

REFERENCES

(1) Wang, H. H., Isaacs, F. J., Carr, P. A., Sun, Z. Z., Xu, G., Forest, C. R., and Church, G. M. (2009) Programming cells by multiplex genome engineering and accelerated evolution. *Nature 460*, 894−898.
(2) Costantino, N., and Court, D. L. (2003) Enhanced levels of lambda Red-mediated recombinants in mismatch repair mutants. *Proc. Natl. Acad. Sci. U. S. A. 100*, 15748−15753.
(3) Ellis, H. M., Yu, D., DiTizio, T., and Court, D. L. (2001) High efficiency mutagenesis, repair, and engineering of chromosomal DNA using single-stranded oligonucleotides. *Proc. Natl. Acad. Sci. U. S. A. 98*, 6742−6746.
(4) Sharan, S. K., Thomason, L. C., Kuznetsov, S. G., and Court, D. L. (2009) Recombineering: a homologous recombination-based method of genetic engineering. *Nat. Protoc. 4*, 206−223.
(5) Yu, D., Ellis, H. M., Lee, E.-C., Jenkins, N. A., Copeland, N. G., and Court, D. L. (2000) An efficient recombination system for chromosome engineering in Escherichia coli. *Proc. Natl. Acad. Sci. U. S. A. 97*, 5978−5983.
(6) Gallagher, R. R., Li, Z., Lewis, A. O., and Isaacs, F. J. (2014) Rapid editing and evolution of bacterial genomes using libraries of synthetic DNA. *Nat. Protoc. 9*, 2301−2316.
(7) Isaacs, F. J. (2011) Precise Manipulation of Chromosomes in Vivo Enables Genome-Wide Codon Replacement. *Science 333*, 348−353.
(8) Lajoie, M. J., Kosuri, S., Mosberg, J. A., Gregg, C. J., Zhang, D., and Church, G. M. (2013) Probing the Limits of Genetic Recoding in Essential Genes. *Science 342*, 361−363.
(9) Mandell, D. J., Lajoie, M. J., Mee, M. T., Takeuchi, R., Kuznetsov, G., Norville, J. E., Gregg, C. J., Stoddard, B. L., and Church, G. M. (2015) Biocontainment of genetically modified organisms by synthetic protein design. *Nature 518*, 55−60.
(10) Rovner, A. J., Haimovich, A. D., Katz, S. R., Li, Z., Grome, M. W., Gassaway, B. M., Amiram, M., Patel, J. R., Gallagher, R. R., Rinehart, J., and Isaacs, F. J. (2015) Recoded organisms engineered to depend on synthetic amino acids. *Nature 518*, 89−93.
(11) Densmore, D. M., and Bhatia, S. (2014) Bio-design automation: software + biology + robots. *Trends Biotechnol. 32*, 111−113.
(12) Ma, N. J., Moonan, D. W., and Isaacs, F. J. (2014) Precise manipulation of bacterial chromosomes by conjugative assembly genome engineering. *Nat. Protoc. 9*, 2285−2300.
(13) Pósfai, G., Plunkett, G., Fehér, T., Frisch, D., Keil, G. M., Umenhoffer, K., Kolisnychenko, V., Stahl, B., Sharma, S. S., de Arruda, M., Burland, V., Harcum, S. W., and Blattner, F. R. (2006) Emergent Properties of Reduced-Genome Escherichia coli. *Science 312*, 1044−1046.
(14) Lorenz, R., Bernhart, S. H., zu Siederdissen, C. H., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011) ViennaRNA Package 2.0. *Algorithms Mol. Biol. 6*, 26.
(15) Wang, H. H., Church, G. M., and Voigt, C. (2011) *Methods Enzymol. 498*, 409−426.
(16) Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. L. (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics 25*, 1422−1423.
(17) Ham, T. S., Dmytriv, Z., Plahar, H., Chen, J., Hillson, N. J., and Keasling, J. D. (2012) Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. *Nucleic Acids Res. 40*, e141−e141.

DOI: 10.1021/acssynbio.5b00219
ACS Synth. Biol. XXXX, XXX, XXX−XXX