Error-guided Search for Part Assignment in Genetic Circuits

Tammy Qiu
Dept of Computer Science
Boston University
tqiu@bu.edu

Swapnil Bhatia
Dept of Electrical and Computer Engineering
Boston University
bhatia.swapnil@gmail.com

Abstract

The design of genetic circuits using modular genetic parts is a canonical approach in synthetic biology. This design approach diverges from conventional electronic design because an optimal set of genetic parts must be chosen to implement the design. Previous work in this area uses standard optimization algorithms to find a permutation of genetic parts that will yield a high quality genetic circuit. For example, the Cello design tool uses simulated annealing to assign parts to a circuit. Here, we explore a new error-guided approach to the problem of part assignment. Briefly, the approach propagates the error between the observed and target behavior of a circuit with a candidate part assignment back to the gradients of each of the parameters of the parts used in the circuit, and uses this gradient to find an improved part assignment. We test this approach against the extant approach and show that it is promising for assigning parts to larger, more complex circuits. The algorithm appears to be scalable to circuits with at least 6-inputs with up to 60 gates.

Cells can navigate complex biochemical environments, communicate with each other to perform complex tasks, and react to specific signals. Because of their ability to control gene expressions, genetic circuits may be designed as the "control plane" in many synthetic biology applications [2]. One of the unique challenges in creating genetic circuits is the design of functional circuits from combinations of parts that may have never been tested together before. To create a genetic circuit, the abstract circuit implementing the desired logic must be synthesized and a set of genetic parts must be found to implement the circuit. For a given logic circuit containing n gates and a part library containing m parts, there are $m!/(m-n)! \in O(m^n)$ possible assignments of parts to gates in the circuit, which grows to trillions of possibilities even with modest library and circuit sizes [5]. Once designed, the circuits must then be assembled and tested in the chassis of interest following an expensive high latency pipeline; therefore it is critical to seed the pipeline with designs with a high chance of success.

The Cello circuit design tool [3] takes as input a specification of a logic circuit and a library of characterized repressors. Cello synthesizes an abstract circuit satisfying the given specification and assigns a set of repressors to the circuit such that the desired logic is realized. Cello uses a simulated annealing algorithm to assign repressors to the circuit and defines a scoring function to aid the search. The search algorithm probabilistically chooses a better candidate solution in the neighborhood of the current solution. Here, we explore the idea of a search algorithm that relates the parameters of the response function of a repressor with the quality of the truth table produced by a given assignment, and uses this relationship to guide the search. Our hypothesis is that information about which repressor and which of its parameters must be changed to obtain a better solution can help guide the search towards high quality solutions. Our preliminary work shows that our hypothesis appears to hold when the circuit to be designed is large and complex. We see evidence that when the circuit to be realized is large and complex an error-guided approach helps arrive at high scoring assignments. We show that for 6-input, 30+ gate circuits, an error-guided approach finds solutions whereas a local search algorithm like simulated annealing is unable to find any solution.

The Approach

The response function of repressor gates can be expressed in the form of a Hill equation:

$$f(x) = y_{min} + \frac{(y_{max} - y_{min})K^n}{x^n + K^n} \tag{1}$$

Each repressor can thus be described with four parameters: K, y_{min}, y_{max}, and n. The response function resembles an "S" curve with the high and low values of the curve used as "on" and "off" states of the signal. The sloping region of the curve is avoided as far as possible in a discrete logic circuit setting. Thus, the inputs to such a gate must fall outside the sloping region to obtain valid signal behavior. Because different repressors may have different response functions, only specific repressor combinations can implement the inputs and outputs of each gate in a circuit. Finding an assignment of repressors to a given logic circuit such that a circuit with robust desired behavior is obtained, is a computationally intractable problem [4]. The quality of a circuit implementation may be captured by measuring how well it represents the desired truth table. Given a desired truth table, any assignment of repressors imposes a signal value to the input and output signals in each row of the truth table. Robust

logic is achieved when the "on" and "off" states of the output are well represented by the output repressor signal. The Cello tool defines a *Range Score* of a repressor assignment as the ratio of the minimum of the on values to the maximum of the off values in the output column of the truth table, thus capturing the worst dynamic range of the implementation.

$$S_{Range} = \frac{min(ON)}{max(OFF)}$$

We define a *Cosine Score* to measure the closeness of the logic achieved by an assignment with the desired logic. The Cosine score is defined as below:

$$S_{Cosine} = \frac{V \cdot T}{||V||||T||} \quad (2)$$

Here, $\cdot$ is the dot product of the real signal vector V from the output column of the truth table from the assignment, and T is the binary output column from the desired truth table. The score ranges between -1 and 1 and represents the cosine of the angle between the output truth table generated by the assignment and the intended truth table.

Error-guided Search

Using the Cosine score, we explore a new approach to searching for repressor assignments. We define each repressor using the four parameters that define its transfer function as described by Eqn. 1. Starting from an initial arbitrary assignment of gates, the algorithm chooses a gate-repressor assignment to change. The choice of which repressor assignment to modify is governed by the relationship between the Cosine score of the assignment, and the parameters of each of the repressors used in the assignment. The algorithm computes partial derivatives of the score with respect to the parameters of each of the repressors and chooses the repressor with the highest impact on the score. The partial derivative of each parameter indicates a direction in which the repressor's response function must change in the parameter space to improve the cosine score. A positive partial derivative indicates the need to replace the current repressor with one whose corresponding parameter is greater in value than the current repressor, while a negative derivative would indicate that the repressor must be replaced by one with a smaller value for that parameter. All repressors are represented as vectors of their parameters. When the repressor for potential replacement is identified, other repressors in the library are ranked in order of their euclidean distance from the repressor marked for replacement, and a candidate replacement is picked based on the sign of the derivative. The replacement is accepted only if it improves the circuit's score. If there are no suitable repressors, then we look at the next highest derivative in the list. We continue this process until we have found a replacement that improves the current score or if there are no more derivatives left to check.

If replacements cannot improve the score, then we explore a repressor swapping strategy for the candidate repressor with the largest derivative. The algorithm then compares the results of the two methods and chooses the one that yields the lowest error. If neither method improves the original circuit, the algorithm moves on to the next largest partial derivative and repeats the process from there. The algorithm

Algorithm 1: One Iteration of Steepest Parameter Update (Chart 1)

Result: Update Circuit Assignment with a new assignment that improves its Error
Let L be the library of available repressors
Let C be a Circuit with an initial set of repressors $C_{assignment}$
Let $E = 1 - S_{cosine}$ for C
Let CurrentError be the Error of C
if *CurrentError* $\geq$ *threshold* **then**
  Let D be the gradient vector of E w.r.t. the circuit's parameters, sorted in descending order
  while D *is not empty and* $CurrentError$ *has not improved* **do**
    Let R be the Repressor with largest derivative $D_{largest}$ in D
    Let R_{sorted} be the list of all other repressors in L sorted in ascending order by Euclidean distance from R
    Let $R_{closest}$ be the repressor that is closest to R where $sign(R_{closest}.param - D.param) = sign(D_{largest})$
    if *such an* $R_{closest}$ *exists* **then**
      Evaluate C after replacing R with $R_{closest}$
      if E *improves* **then**
        Let $ReplacedError$ = $CurrentError$
        Update $C_{ssignment}$ to $ReplacedAssignment$
        Break
  BestSwap ← FindBestSwap(Circuit, R)
  SwapError ← Error of BestSwap
  if *SwapError* < *ReplacedError* **then**
    Let $C_{assignment}$ be BestSwap;

Algorithm 2: Find Best Swap (Chart 2)

Result: Find circuit assignment from swapping all repressors with current R with the best error
Let C be a Circuit with an initial set of repressors $C_{assignment}$;
Let R be a repressor with which every other repressor in the circuit will swap positions;
Let $CurrentError$ be the Error of C;
for *Every Repressor* R_{swap} *in the original assignment that is not* R **do**
  Swap R with R_{swap};
  if *the error of this new assignment is less than the* $CurrentError$ **then**
    return Circuit with new assignment

halts either when it has reached the maximum number of iterations or when the error is below a pre-defined threshold. Algorithms 1 and 2 list the pseudo code for the approach described above. Figure 1 shows an example trajectory of the algorithm on a 3-input circuit.

Results

To evaluate the performance of the proposed algorithm, we performed two sets of experiments. First, we used the circuits reported in the literature [3] and ran our algorithm to generate repressor assignments. Second, we generated larger test circuits and ran our algorithm as well as the Cello assignment algorithm to generate assignments. In our experiments we set the maximum iterations to 1000 and the error threshold to 2×10^{-6}. To prevent the algorithm from stalling at a local optimum, we re-initialize our algorithm 5 times, returning in the end the run that delivered the best Range Score to have a standard metric to compare with circuits reported in the literature.

Three-input circuits

Since we expect our method to perform better on larger and more complex circuits, we report tests on a subset of circuits [3] containing at least 5 gates. For each circuit we made calls to Cello's API with a Verilog description of the circuit, which gave us the best range score out of 5 trajectories. Similarly, with our algorithm we returned the assignment that gave us the best range score out of 5 runs. Since we do not account for toxicity and other part constraints in our algorithm, we also disabled them when testing with Cello. Similarly, to perform a fair comparison we disabled circuit optimizations such as replacing the last repressor with a fluorescent reporter, in Cello.

Table 1 shows the results of our experiments on 3-input circuits. Our algorithm successfully generates range scores that are similar to the ones generated by the Cello algorithm. In some cases, our algorithm finds an improved assignment for the circuit. Notably, the algorithm found these assignments in roughly 1000 evaluations, where an evaluation is creating an assignment and scoring it. In comparison, the Cello algorithm executed at most 1000 iterations but the number of circuit assignments evaluated is not clear. Previous results indicate that a 3-input circuit can require up to 40,000-60,000 evaluations [3]. Our algorithm assigns the same circuits with roughly 1,000 evaluations.

Larger circuits

To evaluate the efficacy of our algorithm on even larger circuits, we first generated an expanded library based off a published library of repressors [3]. After calculating the mean and standard deviation of each parameter across all repressors in the library, we randomly sampled K, y_{min}, y_{max}, and n from a normal distribution to create new hypothetical repressors for testing. For circuits up to 6 inputs, the resulting library was 4 times the size of the existing one but for circuits with more than 6 inputs, we generated a library with up to 200 repressors. Then, we constructed hypothetical abstract circuits by first randomly generating a bit string of length 2^n where n is the number of desired inputs and used it as the output column of the circuit's truth table. Note that a uniformly randomly generated bitstring would, on average, result in boolean functions that require large circuits; thus, the circuits used for testing here are of sufficiently high complexity. Using the truth table, we created a Verilog file and used Cello's API to generate an abstract circuit.

Table 2 shows the results of running the Cello algorithm and our algorithm on larger circuits. We tested the algorithms on 4-6 input circuits comprising 15-60 gates. (In comparison, 3-input circuits contain less than a dozen gates.) Cello's simulated annealing approach performs well up to 5 inputs. For larger circuits, however, the algorithm is unable to find a high scoring solution within the given number of iterations. Our algorithm on the other hand is able to examine a few 100k assignments and still produce a high scoring assignment. This provides support for our hypothesis that an error-guided search can outperform a blind local search algorithm.

Discussion

In this work, we introduce a new approach to the part assignment problem, which is an important problem in the design of synthetic genetic circuits. We introduce a scoring function, the cosine score, to evaluate the quality and correctness of the truth table expected from a design. Similar to gradient-based methods common in machine learning, we develop a new error-guided search algorithm that determines the next local modification to try out based on the partial derivatives of the current solution. We tie the error measured for a design back to the parameters of the repressor transfer function, and use the parameters to find the closest replacement repressor. Our approach matches the assignment correctness and quality scores produced by Cello, a leading gate assignment solver. We also test our algorithm using an expanded synthetic repressor library and show that the algorithm is scalable to larger number of inputs and a larger library. Our work has approached this problem from a purely algorithmic standpoint, setting aside several other design rules related, for example, to toxicity and incompatible part combinations, to illustrate the central idea of our algorithm: we believe that other design principles can be integrated with our approach. Because the proposed approach determines promising search directions as a function of changes in the parameters defining a response function, it may be extensible to an approach that proposes parameters for the *de novo* design of new genetic parts. This may allow the tuning or creation of new genetic parts [1] for higher scoring circuits. While the approach is presented here in the context of repressor circuits, it may be extensible to other part assignment problems with well-defined scoring functions.

References

[1] Ang, J.; Harris, E.; Hussey, B. J.; Kil, R.; and McMillen, D. R. 2013. Tuning response curves for synthetic biology. *ACS synthetic biology* 2(10):547–567.

[2] Brophy, J. A., and Voigt, C. A. 2014. Principles of genetic circuit design. *Nature methods* 11(5):508–520.

Circuit hex	Highest Scoring Cello Assignment	Highest Range score	Highest Scoring Algorithm Assignment	Highest Range score
0x01	HlyIIR H1, AmtR A1, QacR Q2, SrpR S2, BM3R1 B1, PhlF P1	386.895	BM3R1 B3, PsrA R1, LitR L1, QacR Q2, PhlF P1, SrpR S2	467.373
0x04	Amtr A1, PhlF P3, QacR Q1, BM3R1 B3, SrpR S2	467.297	QacR Q1, AmtR A1, BM3R1 B2, PhlF P1, SrpR S2	531.150
0x1c	LitR L1, IcaRA I1, QacR Q1, BetI E1, PhlF P3, SrpR S2	466.421	PhlF P1, LitR L1, BetI E1, BM3R1 B3, QacR Q1, SrpR S2	443.491
0x3d	PhlF P1, LitR L1, HlyIIR H1, QacR Q2, BM3R1 B3, AmtR A1, SrpR S2	462.459	QacR Q1, BM3R1 B2, LitR L1, LmrA N1, BetI E1, PhlF P2, SrpR S3	360.263
0x41	AmtR A1, HlyIIR H1, LitR L1, QacR Q2, BM3R1 B3, PhlF P1, SrpR S2	451.789	PhlF P1, PsrA R1, LitR L1, BetI E1, QacR Q2, BM3R1 B3, SrpR S2	386.444
0x4d	QacR Q1, LitR L1, BM3R1 B1, BetI E1, SrpR S3, PhlF P1	385.917	BM3R1 B2, SrpR S2, HlyIIR H1, AmtR A1, QacR Q2, PhlF P3	324.081
0x60	QacR Q1, BM3R1 B3, LitR L1, BetI E1, PhlF P3, SrpR S2	491.384	IcaRA I1, HlyIIR H1, AmeR F1, BetI E1, PhlF P3, SrpR S2	479.583
0x78	QacR Q1, SrpR S3, LitR L1, BetI E1, PhlF P3	448.660	AmtR A1, LitR L1, BM3R1 B3, PhlF P2, QacR Q1, SrpR S2	368.070
0x81	HlyIIR H1, LitR L1, AmtR A1, QacR Q1, SrpR S3, BM3R1 B2, PhlF P1	382.230	PsrA R1, LitR L1, BetI E1, IcaRA I1, BM3R1 B3, SrpR S2, PhlF P1	324.762
0xbd	BM3R1 B3, LitR L1, PhlF P1, Amtr A1, BetI E1, QacR Q2, SrpR S3	429.067	IcaRA I1, HlyIIR H1, LmrA N1, BetI E1, PhlF P2, BM3R1 B3, SrpR S2	398.773
0xe8	QacR Q1, LitR L1, PsrA R1, AmtR A1, PhlF P3, BM3R1 B3, SrpR S2,	366.055	AmtR A1, QacR Q1, SrpR S2, PsrA R1, LitR L1, BM3R1 B3, PhlF P1	317.709
0xf6	LitR L1, PhlF P1, HlyIIR H1, BetI E1, AmtR A1, BM3R1 B1, QacR Q1, SrpR S2	464.646	LitR L1, AmeR F1, BM3R1 B2, QacR Q2, AmtR A1, BetI E1, PhlF P3, SrpR S2	388.673

Table 1: Comparison of average Range scores by Cello versus average Range scores of the Steepest Parameter Update algorithm for circuits with 5 or more gates.

[3] Nielsen, A. A.; Der, B. S.; Shin, J.; Vaidyanathan, P.; Paralanov, V.; Strychalski, E. A.; Ross, D.; Densmore, D.; and Voigt, C. A. 2016. Genetic circuit design automation. *Science* 352(6281).

[4] Vaidyanathan, P.; Der, B. S.; Bhatia, S.; Roehner, N.; Silva, R.; Voigt, C. A.; and Densmore, D. 2015. A framework for genetic logic synthesis. *Proceedings of the IEEE* 103(11):2196–2207.

[5] Yaman, F.; Bhatia, S.; Adler, A.; Densmore, D.; and Beal, J. 2012. Automated selection of synthetic biology parts for genetic regulatory networks. *ACS synthetic biology* 1(8):332–344.

Circuit hex	**Cello**		**Algorithm**			
	Iterations	Highest Score	Iterations	Circuit Assignments	Differentiation calculations	Highest Score
0x9ebf	1093	3857.439	1000	1000	2923	207.357
0x39a0	343	3791.3432	1000	1000	32462	267.024
0xf88ff71b	421	3692.870	1000	2923	1000	238.121
0xf9f5373c	384	651.870	1000	11338	1000	147.666
0x420ce94 f5e3b163c	179	1.000	372	21536	374	659.523
0xc9d62 0d78e6012	172	1.000	332	466199	332	770.595
0xcf96882 1817b0844	156	1.000	309	309	285123	604.565

Table 2: Comparison between Cello and our algorithm of iterations, number of assignments checked, number of differentiation calculations (for our algorithm) to obtain the highest scoring circuit.

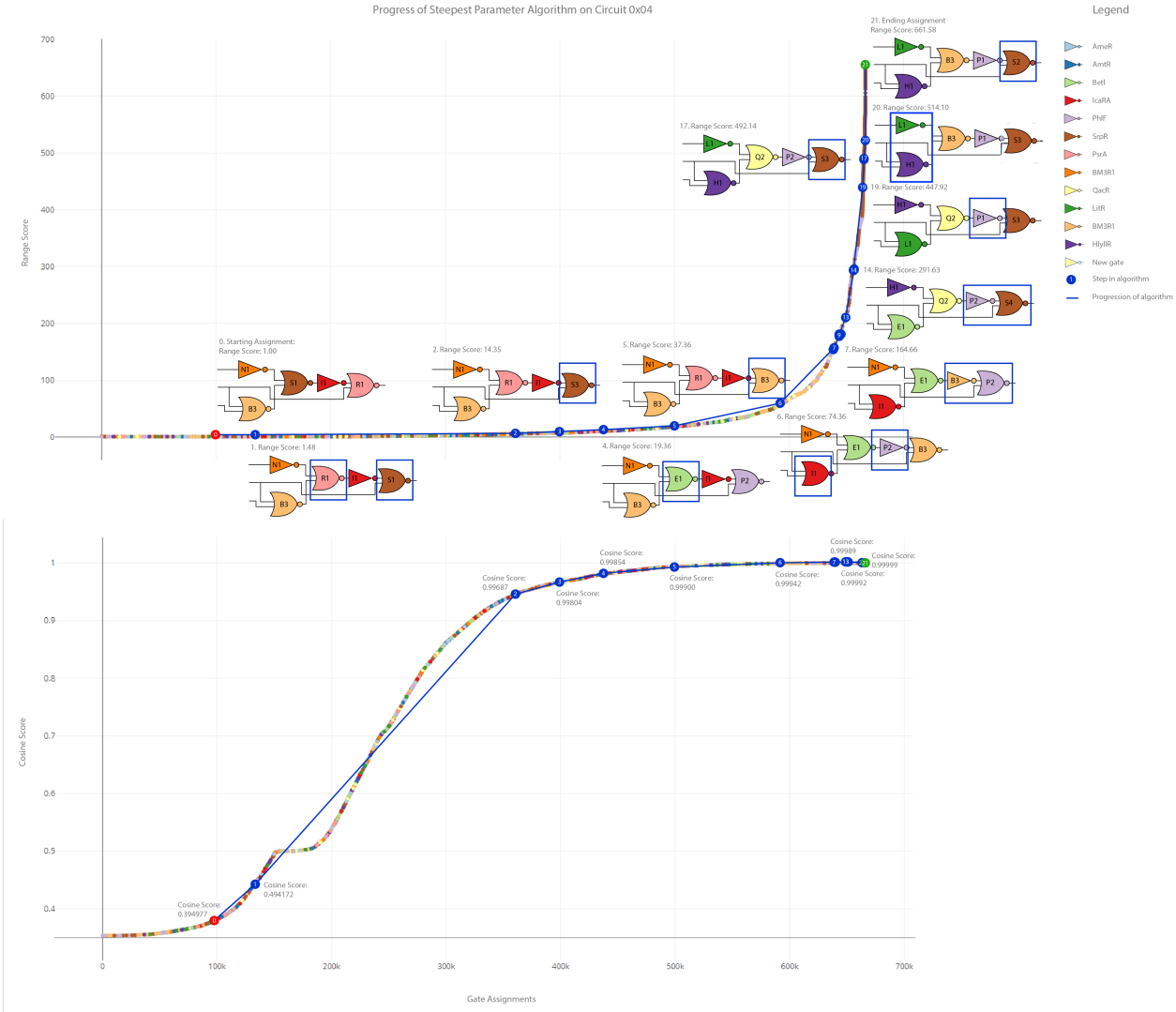

Figure 1: A plot of the algorithm's progression on a sample three-input-circuit. The circuit started with a sub-optimal assignment with a Range score of only 1.000. Notably, while the difference in Range score between steps 1 and 2 is small, it shows a sharp increase in the Cosine score after swapping out SrpR S1 for S3, indicating that the replacement helped orient the output vector into a more Boolean shape. In the following 5 steps, the algorithm makes swaps and replacements that show a gradual growth in score, until step 7 when BM3R1 B3 and PhlF P2 are swapped. With PhlF P2 as the final gate, the score increases by almost 100. By the 14th iteration, the algorithm has already found the best combination of repressors and rearranges the order of the repressors until an optimal score is achieved.