ACS SyntheticBiology

Research Article

pubs.acs.org/synthbio

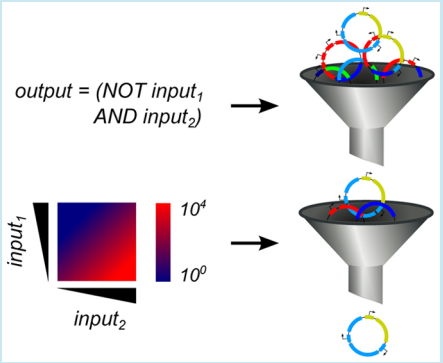
Fast and Accurate Circuit Design Automation through Hierarchical Model Switching

Linh Huynh and Ilias Tagkopoulos*

Department of Computer Science & UC Davis Genome Center, University of California Davis, Davis, California 95616, United States

Supporting Information

ABSTRACT: In computer-aided biological design, the trifecta of characterized part libraries, accurate models and optimal design parameters is crucial for producing reliable designs. As the number of parts and model complexity increase, however, it becomes exponentially more difficult for any optimization method to search the solution space, hence creating a trade-off that hampers efficient design. To address this issue, we present a hierarchical computer-aided design architecture that uses a two-step approach for biological design. First, a simple model of low computational complexity is used to predict circuit behavior and assess candidate circuit branches through branch-and-bound methods. Then, a complex, nonlinear circuit model is used for a fine-grained search of the reduced solution space, thus achieving more accurate results. Evaluation with a benchmark of 11 circuits and a library of 102 experimental designs with known characterization parameters demonstrates a speed-up of 3 orders of magnitude when compared to other design methods that provide optimality guarantees.



KEYWORDS: *biodesign automation, CAD, computer-aided design, module library, part selection, hierarchical model, data-driven model*

The design of biological circuits is a complex, tedious and uncertain process. Over the past decade, research in the area of synthetic biology tries to adopt the engineering principles of standardization, abstraction, model-driven design and automation for a faster, more compatible and reliable outcome. As such, computer-aided design (CAD) is of paramount importance in this exponentially growing field, and it will be essential for synthetic biology to materialize its transformative potential. Over the years there has been considerable progress in the field, although limitations such as scalability, optimality and biological accuracy still need to be addressed.[1−3]

The design of functional components that are part of a larger, interconnected ensemble requires the following fundamental principles to be in place: first, characterized fundamental blocks that can be assembled together into a functional entity should be available; second, the development of predictive models that are accurate enough to capture the dynamic behavior of the designed component; third, access to optimization tools that can support design decisions given user-defined constraints and objective functions, by utilizing the former (fundamental blocks and model predictions) as its inputs. However, there is an inherent trade-off between the second and the third principle. On one hand, capturing fully and accurately the behavior of designed components requires complex models, for example, models that are based on systems of nonlinear differential equations. At the same time, complex models can easily lead to intractable and nonconvex optimization problems. As a result, various CAD tools[4−6] only support circuit simulation without optimization, and even in cases where optimization is supported, it is based on simplistic models, local information and heuristic methods.[7−11] Some of these tools are able to handle modules, two or more parts that are grouped together, which is an important extension as it can significantly lower the construction time.[8,12,13] Still, it remains an open question how the module characterization parameters can be extracted from the full circuit design and the experimental results available in the literature.

This paper introduces a hierarchical two-step approach that balances the model complexity trade-off. Our approach is illustrated in Figure 1. A circuit topology and the desired input−output relationship serve as the input. In the first step, we use a logic-based model to limit the search space to design topologies that can express the desired logic function. Within that reduced search space, we apply a branch-and-bound method to find the set of candidates that minimizes the number of parts and modules. Then, in the second step, a nonlinear model is used to capture the behavior of each candidate circuit and identify the final solution. We used the SBROME framework[8] to implement and measure the performance of this method. To provide the fundamental blocks necessary, we also built a part and module library (PAMLib) that incorporates the set of parts and modules used in ref 12 with characterization

Special Issue: IWBDA 2014

Received: October 16, 2014

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

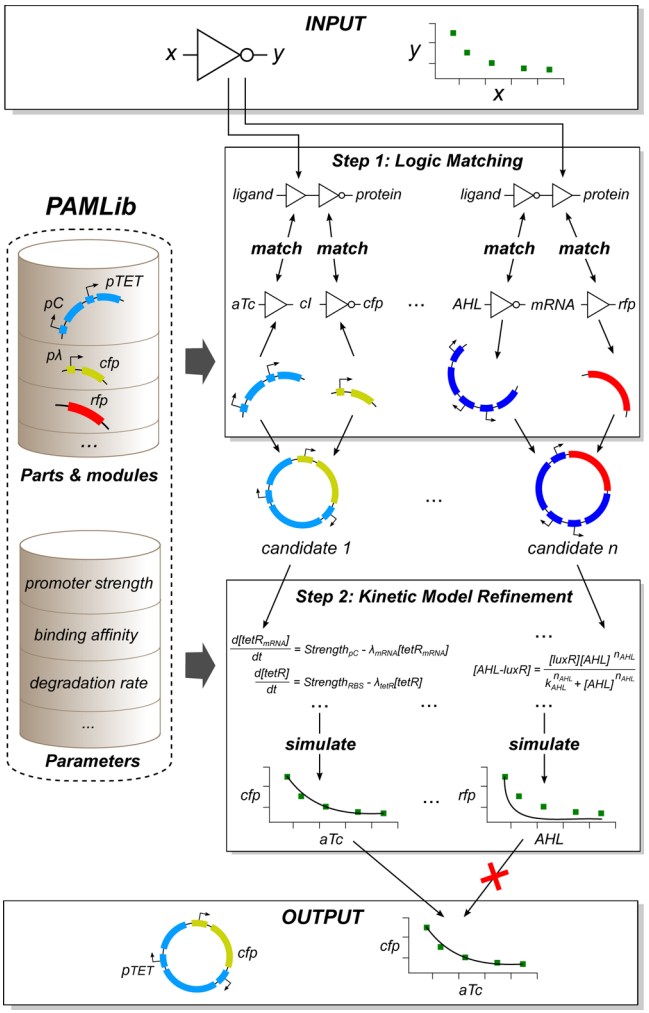


Figure 1. An illustration of the hierarchical model switching. A circuit topology and the desired input−output relationship serve as the input. In the first step, a logical model (represented through a network of logic gates) is used to limit the search space to design topologies that can express the desired logic function. In the second step, a nonlinear model is used to capture the behavior of each candidate circuit and identify the final solution. A part and module library (PAMLib) provides the fundamental blocks necessary for each step. This library incorporates the set of parts and modules (for the first step) with characterization parameters (for the second step). A proposed circuit design and its predicted behavior are provided at the output.

parameters that were extracted from experimentally validated circuits. In the following sections, we present the hierarchical method that we employed and benchmarked against 11 circuits of different sizes (Table S1, Supporting Information).

RESULTS AND DISCUSSION

The general problem that we solve here can be formulated as a multiobjective optimization problem: find the optimal part/module set, given a part library, a circuit topology and an input−output characteristic relationship. A solution can be thought as optimal when it uses the lowest number of blocks (*i.e.*, parts and/or modules) and it has the smallest deviation (measured as the sum of squared errors) from the desired input/output characteristic points. Since these objectives are independent, our approach here only finds Pareto optimal solutions. As described in the Methods section, the core of this approach is a two-step workflow with different granularity levels of the model complexity. In the first step, a logical model that can be derived from the circuit structure (*i.e.*, the interaction of parts) is used. This allows us to apply a branch-and-bound algorithm that branches on the circuit structure while checking on whether other constraints are satisfied. We also implement two extensions that increase the computational performance, a look-ahead technique to exclude subsolutions based on future compatibility issues and a branching technique that can handle complex circuit topologies. In the second step, a nonlinear model is used to capture the circuit behavior with more details. In addition, parameter values are estimated from the high-dimensional data to increase the accuracy of the approach (see Methods). To evalute our framework, we built a benchmark of 11 circuits of various complexity and dynamic behavior.

First, we evaluated how well the number of parts and modules is minimized by using a simple model. The goal here is to minimize the time to construct the circuit from available parts and modules rather than engineering new genetic parts as it was the case in previous work.[14] Therefore, in addition to the minimization of the number of parts and modules needed, we also favor constructs that are in a single plasmid since we may use it directly (*i.e.*, transformation directly into the host cell without amplification and ligation). For that purpose, we define the cost of a part or a module fragment to be one. For a module that is in a single plasmid, the cost is set to $1 - \varepsilon$. Here we set $\varepsilon = 0.01$ so that a solution with that the number of parts and modules still is the dominant factor in cost computation. The solution cost, which is the total cost of parts and modules used in this solution, and the running time for each circuit are showed in Table 1. We also compare to the most parsimonious and fastest method reported until now that employs dynamic programming.[12] We observe a speed-up in running-time between 9 to 4000 in all cases that yield the same solution. In addition, the heuristic approach leads to significantly larger circuits in 9 of the 11 cases, while in the remaining two cases—the feed-forward and the 2-input NOT-AND circuits—the heuristic approach and the simple logical model resulted in the same solution.

Next, we evaluated the performance of the two-step method in the term of how well the final solution approximates the desired circuit dynamics. To predict the behavior of candidates that are determined from the first step, we use a nonlinear model that is based on thermodynamics modeling of the various components, as described in the Methods section and inspired by previous work.[15,16] Table 2 depicts the four binding categories that our model captures and includes both positive and negative regulation for one or two binding sites. Parameter estimation is performed for each module by fitting the corresponding model to the experimental data. As such, a single part that participates in two distinct modules can have characterization parameters that differ, since these have been extracted from different circuits and hence in different contexts. When a new circuit is simulated, the part parameter values are those associated with the circuit/module that the part was extracted and characterized. For the 11 circuits in the benchmark we have extracted a total of 160 parameter values (Table S2, Supporting Information). Table 3 provides a summary of the parameter types and their dynamic range. The simulated output and the desired output of the optimal solution for each circuit is shown in Figure 2. Remarkably, the method is able to generate circuits that approximate adequately the desired dynamic behavior, despite the fact that the parameters for their underlying building blocks (parts,

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

Table 1. A Comparison on the Part and Module Selection between the New Approach (BB) and Other Ones (HS, DP)[a]

		proposed circuit cost		running time (seconds)		
design	input topology size (gate)	HS	BB	HS	DP	BB
2-cascade	2	$2-\varepsilon$	$1-\varepsilon$	1.0×10^{-2}	1.8×10^{-1}	2.0×10^{-2}
3-cascade	3	$3-\varepsilon$	2	1.0×10^{-2}	2.1×10^{-1}	2.0×10^{-2}
4-cascade	4	3	$3-\varepsilon$	1.2×10^{-2}	2.5×10^{-1}	4.0×10^{-2}
band-detector	3	6	5	1.5×10^{-2}	3.4×10^{-1}	8.0×10^{-2}
feed-forward	3	$4-2\varepsilon$	$4-2\varepsilon$	2.6×10^{-2}	6.3×10^{-1}	7.0×10^{-2}
2-not-and	2	2	2	3.1×10^{-2}	6.4×10^{-1}	4.0×10^{-2}
3-input-and	2	3	$3-3\varepsilon$	6.9×10^{-1}	2.3	1.2×10^{-1}
3-not-and	3	$4-\varepsilon$	$4-2\varepsilon$	6.7×10^{-1}	3.6×10^{1}	1.5×10^{-1}
2-to-1-mux	4	7	$5-3\varepsilon$	1.7	1.1×10^{2}	1.6×10^{-1}
D1	4	7	$6-4\varepsilon$	1.1	1.2×10^{3}	9.8×10^{-1}
D2	5	8	$6-4\varepsilon$	2.8	2.0×10^{3}	4.5×10^{-1}

[a]BB, branch and bound; HS, a heuristic search approach; DP, a dynamic programming based approach[12] that enumerates all subsolutions of each node.

Table 2. Models for the Protein Production Rate with Different Promoter Types

# OBS[a]	TF type	Protein production rate Γ	Parameter description	
1	Repressor	$\frac{1}{1+\frac{[R]}{K_1}}F$	$[R]$ K_1 F	Repressor concentration Dissociation constant Maximum protein production rate
	Activator	$\frac{1+f_1\frac{[A]}{K_1}}{1+\frac{[A]}{K_1}}F$	$[A]$ K_1 f_1 F	Activator concentration Dissociation constant Interaction coefficient between TF and RNAP Minimum protein production rate
2	Repressor	$\frac{1}{1+\frac{[R]}{K_1}+\frac{[R]}{K_2}+\frac{\omega[R]^2}{K_1K_2}}F$	$[R]$ K_1, K_2 ω F	Repressor concentration Dissociation constant Interaction coefficient between two repressors Maximum protein production rate
	Activator	$\frac{1+f_1\frac{[A]}{K_1}+f_2\frac{[A]}{K_2}+f_1f_2\frac{\omega[A]^2}{K_1K_2}}{1+\frac{[A]}{K_1}+\frac{[A]}{K_2}+\frac{\omega[A]^2}{K_1K_2}}F$	$[A]$ K_1, K_2 f_1, f_2 ω F	Activator concentration Dissociation constant Interaction coefficient between one activator and RNAP Interaction coefficient between two activators Minimum protein production rate

[a]The number of operator binding sites (OBS) where the transcription factor (TF) can bind.

Table 3. Parameter Extracted and Deposited in the Part and Module Library (PAMLib)

category	parameter	description	unit	quantity	dynamic range
Protein production	F	Protein production rate without transcription factor binding	μM min^{-1}	36	10^5
	f_1	Interaction coefficient between a transcription factor and RNAP	n/a	14	25
	K_1	Dissociation constant	μM	26	10^6
	K_2	Dissociation constant	μM	8	10^5
	ω	Interaction coefficient between two repressors	n/a	8	10^4
Ligand	n	Hill coefficient	n/a	11	4
	K_d	Dissociation constant	μM	11	10^4
Protein	λ	Degradation rate	min^{-1}	41	10^3
	K	Dissociation constant	μM	5	10^6

modules) were inferred from circuits with different input/output specifications. The Pearson correlation coefficient (PCC) for the 11 designs vary from 0.94 to 0.99 with a mean PCC of 0.97. The 2-NOT-AND circuit exhibits the

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

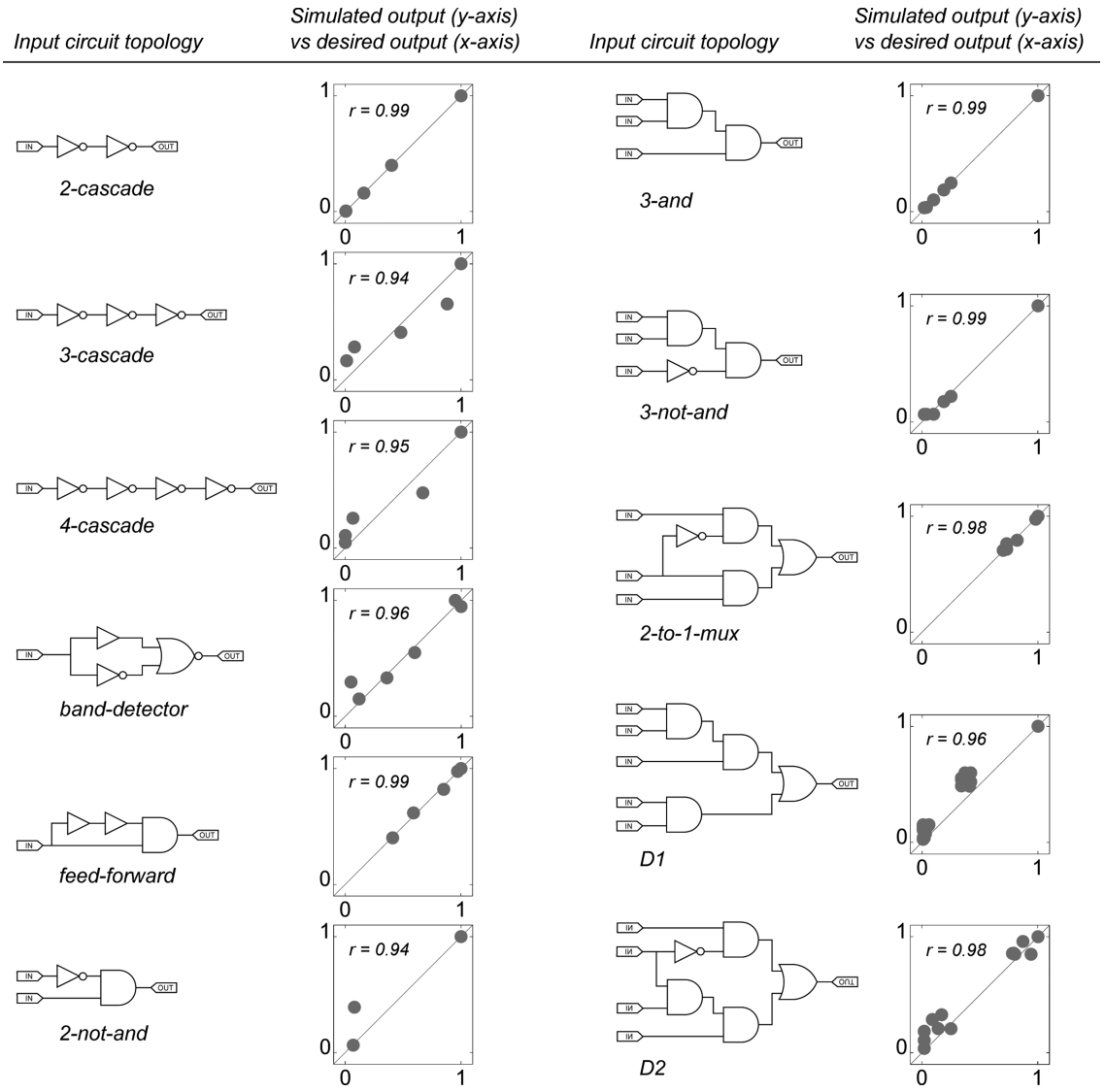


Figure 2. Predicted behavior and desired behavior of each circuit in the benchmark. Each data point represents the normalized output value (the simulated one by the *y*-axis and the desired one by the *x*-axis) of an input value combination.

highest deviation (PCC of 0.94). We further investigated that the source of this discrepancy is not the existence of a better solution (we confirmed that the method selected the optimal solution by exhaustively testing all possible solutions), but the limited number of parts and modules in PAMLib that does not contain parts that could lead to a more favorable combination. In addition, sensitivity analysis argues for the robustness of the results, as the final solution remains unchanged in all cases when a 5% perturbation is added to the parameter set (Figure S1, Supporting Information).

There are several directions that can lead to further improvements. First, an extension to this work would be the integration of the constraint checking into the branch-and-bound step to prune branches as soon as these constraints are violated. Second, the cost function can be reformulated to take into account additional features such as operon optimization, technique-specific assembly constraints[17] (*e.g.*, different cost-functions for Gibson assembly or Golden-Gate). Third, the methodology can be extended to use a linear model,[18,19] instead of a logic-based one as a first step. This will facilitate the incorporation of genetic elements that have a common background but harbor variations that can alter their dynamic behavior, such as promoter libraries that have been generated either through rational design or random mutagenesis of the promoter region.[20] As such, the branch-and-bound scheme that was presented here can be applied for the variant selection problem by grouping subsolutions that have similar behaviors into classes and thus reducing the solution space that the algorithm has to explore. In addition, the method can be extended to handle general circuit topologies that encompass multiple outputs and cyclic graphs. In terms of usability, this implementation was focused more on optimized computation to demonstrate the difference in performance and less so on an intuitive presentation and standard packaging of the resulting software and library. As such, the module library can be repackaged in a standardized way to facilitate its reuse and expansion through active standards, such as the Synthetic Biology Open Language (SBOL).[21,22]

Ultimately, a genome-scale simulator[23−25] can be used to predict the dynamic circuit behavior within a specific host strain and environmental conditions.[26] The triplet of circuit, host strain and environmental conditions can be used to generate a

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

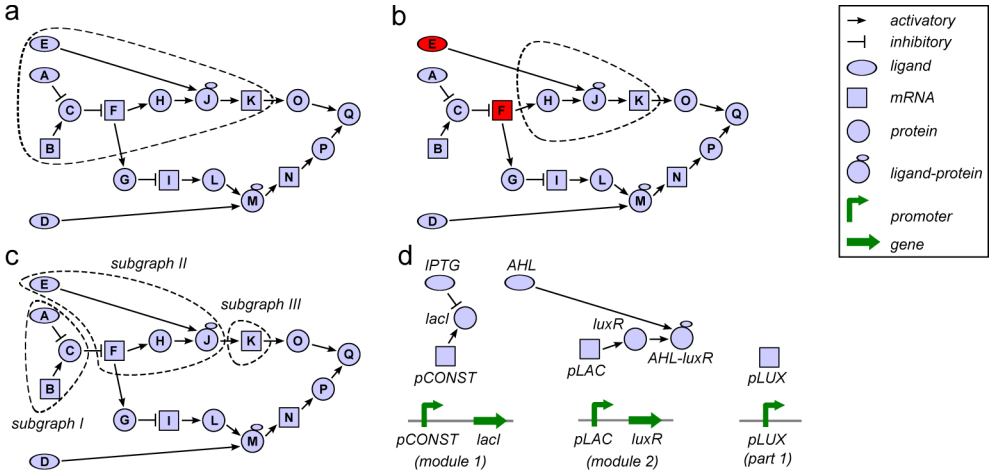


Figure 3. Illustration of definitions on a circuit graph. (a) A transitive input subgraph of node K (marked by the dash line). (b) A transitive fan-in subgraph of node K with corresponding fan-in nodes E and F. (c) An abstract subsolution of node K with 3 subgraphs. (d) A set of one part and two modules and their circuit graph representation, the abstract subsolution in (c) together with this set and the mapping (subgraph I→module 1; subgraph II→module 2; subgraph III→part 1) can form a subsolution of node K.

model of transcription, translation, signal transduction and metabolism. Although the incorporation of a genome-scale model to a design pipeline seems straightforward, there are a number of points to be considered. First, genome-scale and circuit models use a very different approach to modeling, with the former relying in a small set of parameters that usually map statistical associations and not biophysical phenomena. In contrast, circuit models try to capture, in detail, the biophysical dynamics and use the appropriate kinetic constants (*e.g.*, dissociation constants, protein degradation rates) and modeling framework to do so.[27] Merging these two worlds under a unifying framework that increases the model's predictive ability is quite challenging. As both fields move forward, data availability and coordinated efforts in both disciplines will be instrumental to close this gap.[28]

METHODS

Definitions. *Circuit Graph.* We follow[8] in the definition of circuit graph. This graph represents all biological processes (transcription, translation and binding) of a circuit. More specifically, a directed graph represents a circuit, with every node and edge represent the molecular species (ligand, mRNA, protein and their complex) and regulatory relationship (activation, repression), respectively. When only the type of a node (*e.g.*, ligand or protein), but not its name (*e.g.*, IPTG or tetR), is defined, then the graph is called an abstract circuit graph. In this paper, we limit the circuit graph as only a directed acyclic graph with single output node.

Logical Model. A logical model of a circuit graph is a set of logical relationships; each of them is generated from the regulatory relationship at each node by the following simple rules: (i) the binding of an activator to a promoter generates a YES (*i.e.*, amplification) relationship, (ii) the binding of a repressor to a promoter generates a NOT relationship, (iii) the regulation of a hybrid promoter or the binding of two molecular species generate an AND relationship, (iv) the regulation of a tandem promoter generates an OR relationship.

Circuit Graph Matching. A matching between two circuit graphs G_1 and G_2 is a bijection φ between their vertex set such that (i) for each node u in G_1, u and the corresponding node $\varphi(u)$ in G_2 must have the same type, (ii) for each node u in G_1, if the name of both two nodes u and $\varphi(u)$ is defined then these names must be identical, (iii) (u,v) is an edge in G_1 if and only if $(\varphi(u),\varphi(v))$ is an edge in G_2, and (iv) if (u,v) is an edge in G_1 then the type of both two edges (u,v) and $(\varphi(u),\varphi(v))$ must be the same.

Fan-in Node. For a circuit graph, a transitive input subgraph of a node is a subgraph that contains that node and all nodes connected to that node by directed paths (*e.g.*, Figure 3a). A transitive fan-in subgraph of a node is a connected subgraph of the transitive input subgraph of that node that also contains that node itself. Notice that there is only one transitive input subgraph of a node but there is one or many transitive fan-in subgraphs of that node. A fan-in node of a transitive fan-in subgraph is a node outside that subgraph that can connect to another node inside that subgraph by one edge (*e.g.*, Figure 3b).

Part and Module. A part is an elementary genetic element such as a promoter or a gene. A module is a composite part that consists of two parts or more that have been assembled experimentally (*e.g.*, Figure 3d). The biological function of a part or a module is represented by a circuit graph representation above through the transformation as in ref 8. In a circuit graph, a matching between a part (or a module) and a node is a matching (defined above) between the circuit graph representation of that part (or module) and a transitive fan-in subgraph of this node. A fan-in node of this matching is simply a fan-in node of the transitive fan-in subgraph.

Abstract Subsolution and Subsolution. An abstract subsolution of a node is a partition that divides the transitive input subgraph of that node into nonoverlapping subgraphs (*e.g.*, Figure 3c). A subsolution of a node is a triplet of (a) an abstract subsolution of that node where the corresponding set of nonoverlapping subgraphs is $\mathcal{G}$, (b) a set P of parts and modules, and (c) a one-to-one mapping $\sigma:\mathcal{G}\rightarrow P$ such that (i) for each subgraph $G \in \mathcal{G}$, there exists a matching between G and the circuit graph representation of the part (or module) $\sigma(G)$ and (ii) the absence of cross-talk and connection compatibility constraints (*i.e.*, the output molecules of a subcircuit match the input molecules of all connected downstream subcircuits) are satisfied.

For an acyclic circuit graph, the subsolution definition above leads to a recursive relationship: a subsolution of a node is also

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

Algorithm 1: A branch-and-bound based algorithm

Input: An input gene circuit graph G_{in} with an output node v_0 and a library of gene circuit graphs that are classified through a class set $\mathcal{C}$

Output: A list of k candidates, each of them is a set of parts and modules and a corresponding mapping

```
1  begin
       // Step 1:  Estimate the bound on the solution cost
2      for each node v of G_in in the topological order do
3          S[v] = ∅
4          M = Match(G_in, v, C)
5          for each pair (C_i, σ) ∈ M do
6              U = fan_in(σ)
7              if U = ∅ then
8                  cost = cost(α_{C_i})
9                  S[v][cost] = S[v][cost] ∪ {(C_i, σ, ∅)}
10             else
11                 let U = {u_1, u_2, ..., u_k}
12                 for s⃗ = (s_1, s_2, ..., s_k) ∈ ∏_{i=1}^{k} S[u_i] do
13                     cost = cost(α_{C_i}) + Σ_{i=1}^{k} cost(s_i)
14                     S[v][cost] = S[v][cost] ∪ {(C_i, σ, s⃗)}
       // Step 2:  Search for an optimal solution
15     L = ∅
16     X = {cost | S[v_0][cost] ≠ ∅}
17     for cost = min(X) → max(X) do
18         for each reporter protein p do
19             if |L| < k then
20                 L' = search(v_0, cost, p, k − |L|)
21                 L = L ∪ L'
22     return L
```

Algorithm 2: Module classification algorithm

Input: A library (i.e. a set) $\mathcal{G}$ of circuit graphs, each of them represents a part or a module

Output: A set $\mathcal{C}$ of classes $\mathcal{C}_1, \mathcal{C}_2, \ldots$, each class $\mathcal{C}_i$ contains isomorphic circuit graphs where $\alpha_{\mathcal{C}_i}$ denotes the representative graph of the $\mathcal{C}_i$ class

```
1  begin
2      C = ∅
3      for each gene circuit graph G ∈ G do
4          if ∃C_i ∈ C : G ≃ α_{C_i} ∧ cost(G) = cost(α_{C_i}) then
5              C_i = C_i ∪ {G}
6          else
7              C_j = {G}                    // generate a new group
8              α_{C_j} = ExtractAbstractTopology(G)
9              C = C ∪ {C_j}
```

a pair of (a) a part (or a module) together with a matching between this part (or module) and that node, (b) a combination of subsolutions where each of them is of a fan-in node of that matching.

Solution and Cost. The cost of a subsolution is the total cost of all parts and modules used in this subsolution. A solution is simply a subsolution at the output node.

Part and Module Selection Problem. Given an abstract circuit graph and a library of parts and modules, find a solution that has the minimum cost.

A Two-Step Design Workflow. In the first step (Figure 1), all circuit graphs that can express the logic of the input circuit topology are generated and encoded into one, minimal, all-encompassing graph using the method from refs 12, 29. Then a branch-and-bound based algorithm (Algorithm 1) is applied to solve the part and module selection problem (defined above) for this graph. This algorithm is implemented through two steps. In the first step (section Bound Estimation on Solution Cost), the constraints of cross-talk absence and connection compatibility are relaxed to estimate the bound of the cost of all possible solutions. In the second step (section Search for a List of Solutions), with the bound that we have determined, we only need to search for solution for each cost value within that bound in ascending order until a list of k solutions (candidate circuits) are found. In addition, two extensions (sections Extension 1 and Extension 2) are also applied to improve the computational performance. In the second step (Figure 1), we rank each candidate circuit by the deviation from its behavior to the given desired one. The behavior of each candidate circuit is predicted by using a nonlinear thermodynamics model[15,16] with parameter values that are estimated from the experimental characterization data (section Rank Solutions by Their Predicted Behavior). The technical details of each step are presented in the following sections.

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

Bound Estimation on Solution Cost. To estimate a lower bound, we relax the constraints of cross-talk absence and connection compatibility. With this relaxation, the information about the node name in the circuit graph representation becomes irrelevant since we do not have to check cross-talk or compatibility. Therefore, to make the computation more efficient, we group all parts and modules based on their circuit graph topology and their cost. Algorithm 2 summarizes this preprocessing step.

Because of the recursive relationship on the subsolution above, the bound on the subsolution cost of each node can be estimated as in algorithm 1 (step 1). We store both the bound information after bound estimation of each node and all subsolutions with their costs pertaining to that node, so we can efficiently search for a solution in the second step. For that purpose, for each node v, we can use a hash table $\mathcal{S}[v]$ that hashes each abstract subsolution by its cost (*i.e.*, $\mathcal{S}[v][\text{cost}]$ contains all abstract subsolutions of node v, each of them has the same cost). However, this approach makes the computation inefficient since we have to enumerate all such abstract subsolutions. Therefore, we propose a further step to group all abstract subsolutions of a node by the matching between that node with a part/module and the cost distribution of subsolutions at fan-in nodes of this matching. More specifically, $\mathcal{S}[v][\text{cost}]$ contains all triplets $(C_i, \sigma, \vec{s})$ where C_i is an isomorphic class of parts and modules, σ encodes the matching between v and the representing circuit graph α_{C_i} of this class while $\vec{s}$ encodes the cost distribution of subsolutions of fan-in nodes of this matching. By that way, $\mathcal{S}[v]$ is updated iteratively by traversing all nodes of the input circuit graph by the topological order (*i.e.*, if there is an edge from u to v then u will be traversed before v). At each node v, the procedure Match will find all possible pairs (C_i, σ). Then, for each such pair, all cost distributions $\vec{s}$ of subsolutions of fan-in nodes of the matching σ are explored to form a new triplet that will be inserted into $\mathcal{S}[v]$ by hashing the total cost.

Search for a List of Solutions. Recall that a solution is a subsolution of the output node, thus the bound on its cost is determined from the first step. Therefore, we only need to search for solutions for each cost value within that bound in ascending order until a list of k solutions (*i.e.*, candidates) is found as in algorithm 1 (step 2). Notice that the searching in this step is for a complete solution, so the constraints of cross-talk absence and the connection compatibility have been taken into account.

By utilizing the information stored in $\mathcal{S}[v][\text{cost}]$ from the first step, this searching operation can be done recursively from the output node; *i.e.*, we can retrieve all possible matchings between a part or a module and that node from $\mathcal{S}$ and then try for each matching by repeating the search of the subsolution at each fan-in node of this matching. To make the memory usage efficient, we choose depth-first search and concomitantly we eliminate the need for recursion. Since many intermediate searches are repeated, we propose to store their results locally to be reused later on. The auxiliary function Search (Supporting Information) summarizes this procedure.

Extension 1 (Interaction Look-Ahead). As an extension to the previous algorithm, an interaction look-ahead scheme was implemented in conjunction to the branch-and-bound based method to prune subsolution branches. In this case, when the part/module classification is performed, as in Algorithm 2, we also take into account all possible interactions of the input/output molecular species of this part/module (*i.e.*, the molecular species of the input/output node in the circuit graph representation of this part/module) with other molecular species (*i.e.*, the type of those species and the corresponding interaction type). By the input/output interaction information on a part/module and of a node in the abstract circuit graph, we are able to predict if the matching between this node and that part/module will lead to a violation of the connection compatibility constraint or not. If so, we prune all solutions that have resulted from this matching.

Extension 2 (Branching Extension). A branching node is a node that connects to two or more nodes. For example, node F is a branching node of the circuit graph in Figure 3a. In this example, if a final solution (of node Q) that contains a subsolution of node O, a subsolution of node P and a maching between a part/module with node Q then one of these two subsolutions will cover the transitive input subgraph of the branching node F (*i.e.*, subgraph of nodes A, B, C, F) and the other subsolution will not. Here "a subsolution covers a transitive input subgraph" means each node of this transitive input subgraph is included in one of subgraphs that are generated by the partition of the subsolution. Therefore, to be able to handle a circuit graph that contains branching nodes, we have to extend the subsolution definition above by taking into account if a subsolution covers the transitive input subgraph of a particular branching node or not. In this work, this extension is implemented by adding a vector of flags (each flag for one branching node) into each subsolution to mark if it covers the transitive input subgraph of that branching node or not.

Rank Solutions by Their Predicted Behavior. To choose the optimal design among candidates in the second step (Figure 1), we define another cost function f for each candidate C

$$f(C) = \sum_{i \in I} (s_C(i) - d(i))^2$$

where I is a set of inputs, $s_C(i)$ and $d(i)$ are the simulated output value of C for input i and the desired output value for input i respectively. The final design C^* is the one that has the minimum cost $f(C^*)$. We use a thermodynamics model[15,16] to determine the simulated output values of each circuit. More specifically, the transcription and the translation processes are modeled by the following equation

$$\frac{\mathrm{d}[\text{protein}]}{\mathrm{d}t} = \Gamma - \lambda_{\text{protein}}[\text{protein}]$$

where $[\text{protein}]$, λ_{protein}, Γ are the protein concentration, protein degradation rate and the protein production rate, respectively. The protein production rate is modeled for four types of binding that includes both positive and negative regulation with one or two binding site. Table 2 provides the mathematical formulation for each of these cases. For the protein−protein binding, the protein complex concentration is modeled at equilibrium

$$K = \frac{[\text{protein}_1][\text{protein}_2]}{[\text{protein}_1 - \text{protein}_2]}$$

where $[\text{protein}_1]$, $[\text{protein}_2]$ and $[\text{protein}_1 - \text{protein}_2]$ are the concentration of proteins and the protein complex, respectively, while K is the dissociation constant. In the case of ligand−protein binding, we use the Hill equation as follows:

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX

$$[\text{ligand}-\text{protein}] = \frac{[\text{ligand}]^n}{K_d^n + [\text{ligand}]^n}[\text{protein}]$$

where [ligand−protein], [ligand], [protein] are the concentration of the ligand−protein complex, the ligand, and the protein, respectively. The parameters n and K_d correspond to the Hill coefficient and the dissociation constant, respectively.

To estimate the parameter values, we fitted the model and the experimental characterization data of each circuit in the PAMLib. The nonlinear optimization solver[30] was used to find the locally optimal solution for that fitting problem. In total, we estimated 160 parameter values (Table S2, Supporting Information) to simulate the behavior of candidate designs for all 11 circuits in our benchmark set.

All related data are available as a resource at http://tagkopouloslab.ucdavis.edu/software.html.

■ ASSOCIATED CONTENT

Supporting Information

This material is available free of charge *via* the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: itagkopoulos@ucdavis.edu.

Author Contributions

L.H. wrote the code and performed the experiments. I.T. conceived the project and supervised all development and analysis. L.H. and I.T. analyzed the data and wrote the paper.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

We would like to acknowledge support from the NSF CAREER Grant #1254205 to IT.

■ REFERENCES

(1) Church, G. M., Elowitz, M. B., Smolke, C. D., Voigt, C. A., and Weiss, R. (2014) Realizing the potential of synthetic biology. *Nat. Rev. Mol. Cell Biol. 15*, 289−294.

(2) Brophy, J. A., and Voigt, C. A. (2014) Principles of genetic circuit design. *Nat. Methods 11*, 508−520.

(3) Slusarczyk, A. L., Lin, A., and Weiss, R. (2012) Foundations for the design and implementation of synthetic genetic circuits. *Nat. Rev. Genet. 13*, 406−420.

(4) Chandran, D., Bergmann, F. T., and Sauro, H. M. (2009) TinkerCell: modular CAD tool for synthetic biology. *J. Biol. Eng. 3*, 19.

(5) Hill, A. D., Tomshine, J. R., Weeding, E. M., Sotiropoulos, V., and Kaznessis, Y. N. (2008) SynBioSS: the synthetic biology modeling suite. *Bioinformatics 24*, 2551−2553.

(6) Marchisio, M. A., and Stelling, J. (2008) Computational design of synthetic gene circuits with composable parts. *Bioinformatics 24*, 1903−1910.

(7) Huynh, L., Kececioglu, J., Köppe, M., and Tagkopoulos, I. (2012) Automatic design of synthetic gene circuits through mixed integer non-linear programming. *PLoS One 7*, e35529.

(8) Huynh, L., Tsoukalas, A., Köppe, M., and Tagkopoulos, I. (2013) SBROME: A scalable optimization and module matching framework for automated biosystems design. *ACS Synth. Biol. 2*, 263−273.

(9) Beal, J., Weiss, R., Densmore, D., Adler, A., Appleton, E., Babb, J., Bhatia, S., Davidsohn, N., Haddock, T., Loyall, J., Schantz, R., Vasilev, V., and Yaman, F. (2012) An end-to-end workflow for engineering of biological networks from high-level specifications. *ACS Synth. Biol. 1*, 317−331.

(10) Rodrigo, G., and Jaramillo, A. (2012) AutoBioCAD: full biodesign automation of genetic circuits. *ACS Synth. Biol. 2*, 230−236.

(11) Myers, C. J., Barker, N., Jones, K., Kuwahara, H., Madsen, C., and Nguyen, N.-P. D. (2009) iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics 25*, 2848−2849.

(12) Huynh, L., and Tagkopoulos, I. (2014) Optimal part and module selection for synthetic gene circuit design automation. *ACS Synth. Biol. 3*, 556−564.

(13) Roehner, N., and Myers, C. J. (2014) Directed acyclic graph-based technology mapping of genetic circuit models. *ACS Synth. Biol. 3*, 543−555.

(14) Marchisio, M. A., and Stelling, J. (2011) Automatic design of digital synthetic gene circuits. *PLoS Comput. Biol. 7*, e1001083.

(15) Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J., and Phillips, R. (2005) Transcriptional regulation by the numbers: models. *Curr. Opin. Genet. Dev. 15*, 116−124.

(16) Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J., Kuhlman, T., and Phillips, R. (2005) Transcriptional regulation by the numbers: applications. *Curr. Opin. Genet. Dev. 15*, 125−135.

(17) Appleton, E., Tao, J., Haddock, T., and Densmore, D. (2014) Interactive assembly algorithms for molecular cloning. *Nat. Methods 11*, 657−662.

(18) Davidsohn, N., Beal, J., Kiani, S., Adler, A., Yaman, F., Li, Y., Xie, Z., and Weiss, R. (2015) Accurate predictions of genetic circuit behavior from part characterization and modular composition. *ACS Synth. Biol.*, DOI: 10.1021/sb500263b.

(19) Rothschild, D., Dekel, E., Hausser, J., Bren, A., Aidelberg, G., Szekely, P., and Alon, U. (2014) Linear superposition and prediction of bacterial promoter activity dynamics in complex conditions. *PLoS Comput. Biol. 10*, e1003602.

(20) Ellis, T., Wang, X., and Collins, J. J. (2009) Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat. Biotechnol. 27*, 465−471.

(21) Galdzicki, M., et al. (2014)*et al.* The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol. 32*, 545−550.

(22) Roehner, N., Oberortner, E., Pocock, M., Beal, J., Clancy, K., Madsen, C., Misirli, G., Wipat, A., Sauro, H., and Myers, C. J. (2015) Proposed data model for the next version of the Synthetic Biology Open Language. *ACS Synth. Biol. 4*, 57−71.

(23) Carrera, J., Estrela, R., Luo, J., Rai, N., Tsoukalas, A., and Tagkopoulos, I. (2014) An integrative, multi-scale, genome-wide model reveals the phenotypic landscape of *Escherichia coli*. *Mol. Syst. Biol. 10*, 735.

(24) Liu, J. K., O'Brien, E. J., Lerman, J. A., Zengler, K., Palsson, B. O., and Feist, A. M. (2014) Reconstruction and modeling protein translocation and compartmentalization in *Escherichia coli* at the genome-scale. *BMC Syst. Biol. 8*, 110.

(25) Karr, J. R., Sanghvi, J. C., Macklin, D. N., Gutschow, M. V., Jacobs, J. M., Bolival, B., Jr, Assad-Garcia, N., Glass, J. I., and Covert, M. W. (2012) A whole-cell computational model predicts phenotype from genotype. *Cell 150*, 389−401.

(26) Kim, M., Zorraquino, V., and Tagkopoulos, I. (2015) Microbial Forensics: Predicting Phenotypic Characteristics and Environmental Conditions from Large-Scale Gene Expression Profiles. *PLoS Comput. Biol.*, DOI: 10.1371/journal.pcbi.1004127.

(27) Kirk, P., Thorne, T., and Stumpf, M. P. (2013) Model selection in systems and synthetic biology. *Curr. Opin. Biotechnol. 24*, 767−774.

(28) Way, J. C., Collins, J. J., Keasling, J. D., and Silver, P. A. (2014) Integrating biological redesign: where synthetic biology came from and where it needs to go. *Cell 157*, 151−161.

(29) Lehman, E., Watanabe, Y., Grodstein, J., and Harkness, H. (1997) Logic decomposition during technology mapping. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 16*, 813−834.

(30) Johnson, S. G. The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt, accessed Feb. 15, 2015.

DOI: 10.1021/sb500339k
ACS Synth. Biol. XXXX, XXX, XXX−XXX