ACS SyntheticBiology

Research Article

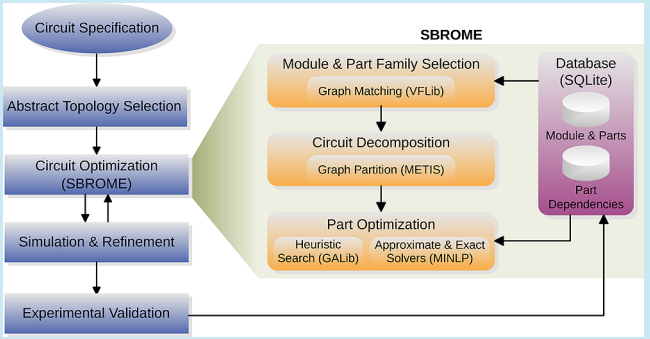
pubs.acs.org/synthbio

SBROME: A Scalable Optimization and Module Matching Framework for Automated Biosystems Design

Linh Huynh,[†] Athanasios Tsoukalas,[†] Matthias Köppe,[‡] and Ilias Tagkopoulos*[,†]

[†]Department of Computer Science and UC Davis Genome Center and [‡]Department of Mathematics, University of California, Davis, California 95616 United States

ABSTRACT: The development of a scalable framework for biodesign automation is a formidable challenge given the expected increase in part availability and the ever-growing complexity of synthetic circuits. To allow for (a) the use of previously constructed and characterized circuits or modules and (b) the implementation of designs that can scale up to hundreds of nodes, we here propose a divide-and-conquer **S**ynthetic **B**iology **R**eusable **O**ptimization **Me**thodology (SBROME). An abstract user-defined circuit is first transformed and matched against a module database that incorporates circuits that have previously been experimentally characterized. Then the resulting circuit is decomposed to subcircuits that are populated with the set of parts that best approximate the desired function. Finally, all subcircuits are subsequently characterized and deposited back to the module database for future reuse. We successfully applied SBROME toward two alternative designs of a modular 3-input multiplexer that utilize pre-existing logic gates and characterized biological parts.



KEYWORDS: *biodesign automation, CAD tool, computer-aided design, modular design, module matching, part selection, multiplexer*

One of the major challenges in synthetic biology is the development of methodologies that are scalable with respect to the number of parts and circuit complexity. In addition, it is highly desired to reuse previously constructed synthetic circuits wherever possible due to the time-consuming nature of any bioengineering effort.[1−3] Although this may lead to larger and less fine-tuned circuits compared to those that are designed *ab initio*, the increase in design compatibility and the significant decrease of the time-to-market outweighs these shortcomings. Moreover, the application of engineering principles to bioengineering will considerably enhance our ability to design, control, and understand biological systems in general and gene regulatory circuits specifically. In the past decade, a number of groups have introduced modular circuits that have been reused successfully to design systems of higher complexity.[4−8] This year, an experimental methodology for modular design was introduced, and its utility was demonstrated by reconfiguring a genetic toggle switch[9] to produce feed-forward loops.[10] In the computational realm, there has been a similar effort to support the efficient simulation of modular structures. For example, the process modeling tool PROMOT allows the user to define modular structures so that pre-existing models can be reused.[11] The modular model definition language Antimony employs similar concepts to capture modularity in design,[12] and a hierarchical modeling approach was recently introduced from the same group.[13] The GEC language goes a step further by enabling users to define modules with abstract parts, such as repressors and activators.[14] The Eugene specification language[15] operates in a similar fashion to define gene circuits that are then compiled to meet the user specifications. These efforts provide the language level formalism necessary for the future development of synthetic biology computer-aided design (CAD) tools, which have evolved considerably from early approaches[16] and have already achieved notable advances (a recent review of the CAD tools available[17]). An end-to-end framework called TASBE[18] has also been introduced recently and aspires to provide an efficient pipeline to computationally driven synthetic circuit design.

Despite the several advances in the field, there is still a long road ahead to create CAD-based solutions that are of high value to experimentalists. Currently few tools support optimization methods for automated circuit design (e.g., see refs 19 and 20) that are based on heuristics and hence cannot guarantee optimality or provide bounds for the proposed solution.[21−23] Similarly, with the exception of a method published earlier this year,[24] current tools do not provide an automatic matching method for reusable designs. Furthermore, as more parts become available, there will be an increasing need of algorithms that are scalable both in time and space. To bridge this gap, we introduce a **S**ynthetic **B**iology **R**eusable **O**ptimization **Me**thodology (SBROME) that can scale well to circuits with dozens to hundreds of parts. Toward this goal, we first curated the synthetic biology literature and built an online module database with circuits with two or more components. Then, we built a relational

Special Issue: IWBDA 2012

Received: September 23, 2012
Published: February 26, 2013

part database with part characterization information that can be used for synthetic circuit design. Special emphasis was given to libraries of promoter variants that have been quantitatively characterized and can be alternatively selected by the optimization tool to achieve the desired expression level. The SBROME framework first uses graph isomorphism algorithms to match modules to topology and then decompose the resulting topology to subcircuits that can be efficiently solved through exact or approximate optimization methods. Its methods include graph-theoretic approaches that allow both abstract and specific definitions of parts and modules. This allows multiple architectures to be considered as possible solutions and can accommodate the diversity that is present in biological systems (summarized in ref 25 and extended in Table 1).

Table 1. Node Types and Names That Are Currently Available in SBROME

node type T_V	node name L_V	refs
molecular species		
ligand	IPTG, aTc, L-arabinose, Mg2+, Sal, 3OC6-HSL, 3OC12-HSL, C4-HSL, nalidixic acid, DOX	
protein	lacI, tetR, araC, cI, luxR, luxI, lasR, lasI, rhlR, rhlI, lexA, ntrC, lacZ, gfp, yfp, rfp, cI434, hrpR, hrpS, nahR	
RNA	pCONST, pLAC, pTET, pBAD, pLAMBDA, pLUX, pLAS, pRHL, phrpL, pmgrB, pT7, pSAL, pSOS	
ligand−protein complex	L-arabinose-araC, 3OC6-HSL-luxR, 3OC12-HSL-lasR, C4-HSL-rhlR	
RNA complex	taRNA-crRNA	
protein complex	hrpR-hrpS	
functional nodes		
YES-gate	pLAC-gfp, pBAD-gfp, pLUX-gfp	5
	pTET-yfp	47
	pBAD-cI-p$O_R O_{lac}$-gfp	48
NOT-gate	$p_{Ltet-O1}$-luxR-pOmpC-cI-$p_{lux-lambda}$-lacZ	49
	pLAC-cI-pLAMBDA-gfp	5
	pTET-lacI-pLAC-yfp	47
AND-gate	pSAL-supD-pBAD-T7ptag-pT7-gfp, pLUX-supD-pmgrB-T7ptag-pT7-gfp, pSAL-supD-pmgrB-T7ptag-pT7-gfp	4
	pLAC-hrpR-pBAD-hrpS-phrpL-gfp, pLUX-hrpR-pBAD-hrpS-phrpL-gfp	5
	pTET-luxR-pLUX-gfp	50
	pbla-LuxR-pCONST-lacI-$p_{luxI-lacO}$-gfp	51
NOR-gate	pBAD-pTET-lasI, pBAD-pTET-yfp, pBAD-pLAS-rhlI, pBAD-pLAS-yfp, pTET-pLAS-rhlI, pTET-pLAS-yfp	6
oscillator	pLAC-tetR-pTET-cI-pLAMBDA-lacI	52
	$p_{lac/ara}$-lacI-$p_{lac/ara}$-araC-yfp	53
	glnAp2-glnG-glnKp-lacI	54

METHODS

An Overview of the Integrated SBROME Framework. In ref 26, we have decomposed the problem of synthetic circuit design in three independent phases: (a) identification of abstract network topology, (b) assignment of proteins and promoters to the abstract network topology to create a fixed network topology, and (c) assignment of optimal mutants for the proteins and promoters that were selected in phases one and two to create the final specific network topology. In a recent paper,[27] we provided an exact optimization algorithm for the subproblem of optimal part selection once the circuit topology has been specified (phase 3). Here, we adhere to the same three-phase framework and present a novel approach to address the problem of assigning specific promoters and proteins given an abstract network topology (phase 2). In addition, we describe and evaluate a circuit decomposition method to facilitate the application of optimal part variant selection (phase 3) with exact and approximate methods.

Figure 1 illustrates the proposed "divide-and-conquer" approach. In the first step of the SBROME platform, a module library with preconstructed designs is queried to find possible partial or full matches to the given abstract circuit topology, which has been specified by the user. To do so, both module and abstract circuit networks have to be transformed in an equivalent graph representation that can support the complex relationships and diverse dependencies that are encoded within a synthetic circuit diagram. After we match existing modules to the transformed network, we decompose the circuit into subcircuits of relatively small size (4−6 components) by applying graph partitioning algorithms that minimize interconnectivity among circuit partitions. The link values are subsequently quantized, which reduces the problem dimensionality and allows for exact methods to be applied efficiently in order to find the part combination that minimizes the difference between the desired and actual gene expression values. Finally, the computationally inferred subcircuits are experimentally characterized and then integrated to comprise the final construct. All newly characterized circuits are deposited in the module database for future use.

Graph Representation and Transformation (Figure 2). The user-defined input network may consist of abstract topological features that can be either elementary part types or generalized devices. Under elementary part types, we incorporate genes, promoters, and ribosome binding sites, while amplifiers, oscillators, and logic gates are the generalized devices that are currently considered in SBROME. Network nodes that represent these generalized devices are called *functional nodes* and can be either specific or abstract, depending on whether they map for a specific physical topology. For example, a user has the option to specify that a node is an AND gate without any other constraints on its topology and underlying parts (abstract functional node) or a specific AND gate design that has been previously published (specific functional node). This option allows the user to select the level of abstraction that is optimal for the application at hand. Once the user has specified the initial input network, SBROME transforms it to a standard graph so that graph matching and isomorphism algorithms can be systematically applied. The final transformed graph consists of only three node types (namely, mRNAs, proteins, and ligands), their combinations, and any functional nodes that are yet to be decomposed into parts. The edges in the directed graph correspond in one of the following categories:

- molecular binding: binding of a molecular species to another, e.g., a ligand binds to a protein to make a ligand−protein complex. Since both molecular species have to be present for this reaction to occur, the relationship that this link encodes is similar to an AND function.
- transcription: The contribution of transcription factors to mRNA can be either activatory or inhibitory, depending on the role of these proteins during transcription. Edges of this type capture an AND relationship during graph transformation (in the case of repressors, a negation of the input precedes the AND integration)

dx.doi.org/10.1021/sb300095m | ACS Synth. Biol. 2013, 2, 263−273

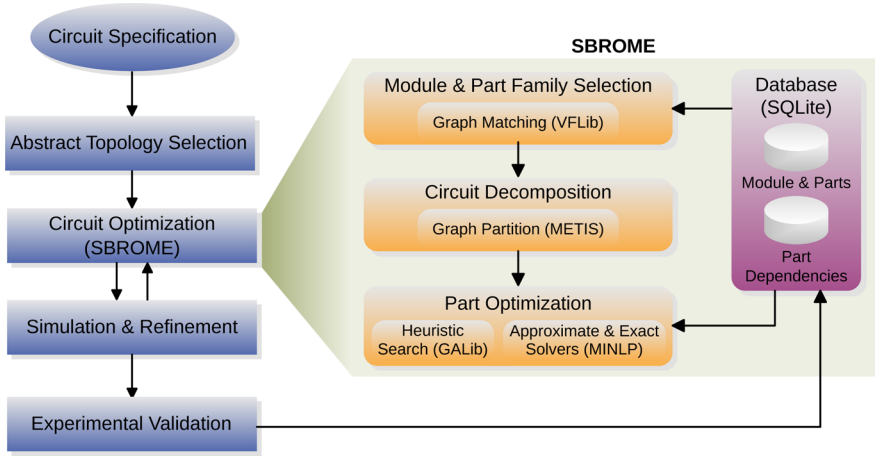


Figure 1. Workflow overview of the synthetic circuit design framework. User circuit specifications and an abstract circuit topology serve as inputs to the SBROME platform that produces a fully defined circuit, which is further used for simulation and refinement. SBROME can itself be decomposed into three sequential steps. In the *module and part family selection* step, already built modules and part types are matched to the abstract circuit. Then the circuit is decomposed in the *circuit decomposition* step by using standard graph partitioning algorithms. In the last *part optimization* step, the specific part variants are being chosen so that the circuit behavior best approximates the desired behavior that has been specified by the user.

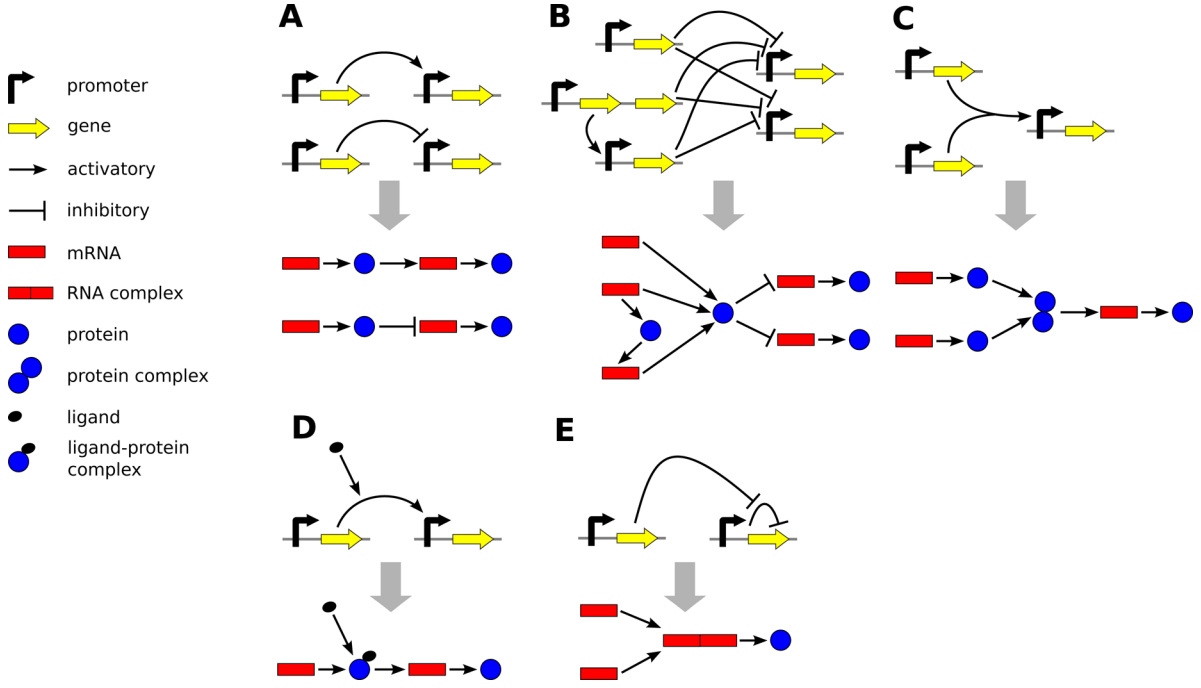


Figure 2. Gene circuit graph representation and transformation. (A) Protein−DNA interaction. (B) Protein−DNA interaction in cases where more than one gene copy exists. (C) Protein−protein interaction. (D) Ligand−protein interaction, where only the active form of the protein is shown. (E) RNA−RNA interaction.

• translation: The contribution of mRNA to its respective protein is always positive. In the presence of multiple gene copies that contribute to the same protein pool, this captures an additive relationship.

Although the edges are treated equally during the graph matching step, these categories are important at the simulation step as they correspond to processes that are modeled differently (see Gene Expression Model section). The same transformation rules apply to the modules that comprise the module database.

A more formal treatment of the graph transformation is given below. The circuit specification that the user supplies to SBROME includes an abstract circuit topology and a desired behavior that is an input-output relationship under steady state conditions. Since both the inputs and the outputs of biological circuits are usually molecular species (for example, inputs can be ligands and outputs can be reporter proteins), we can represent their interactions as a directed graph in which each node represents a molecular species and each edge represents an interaction between the source and target nodes. Formally, a circuit graph is defined as in the following:

Definition 1. For given sets of node types T_V, node names L_V, and edge types T_E. A circuit graph is a 7-tuple $G = (V, E, V_I, V_O, \tau_1, \tau_2, \nu)$ where

1. V is a finite set of nodes,
2. $E \subset V \times V$ is a set of directed edges without any self-loop,
3. $V_I \subseteq V$ is a set of input nodes,

dx.doi.org/10.1021/sb300095m | *ACS Synth. Biol.* 2013, 2, 263−273

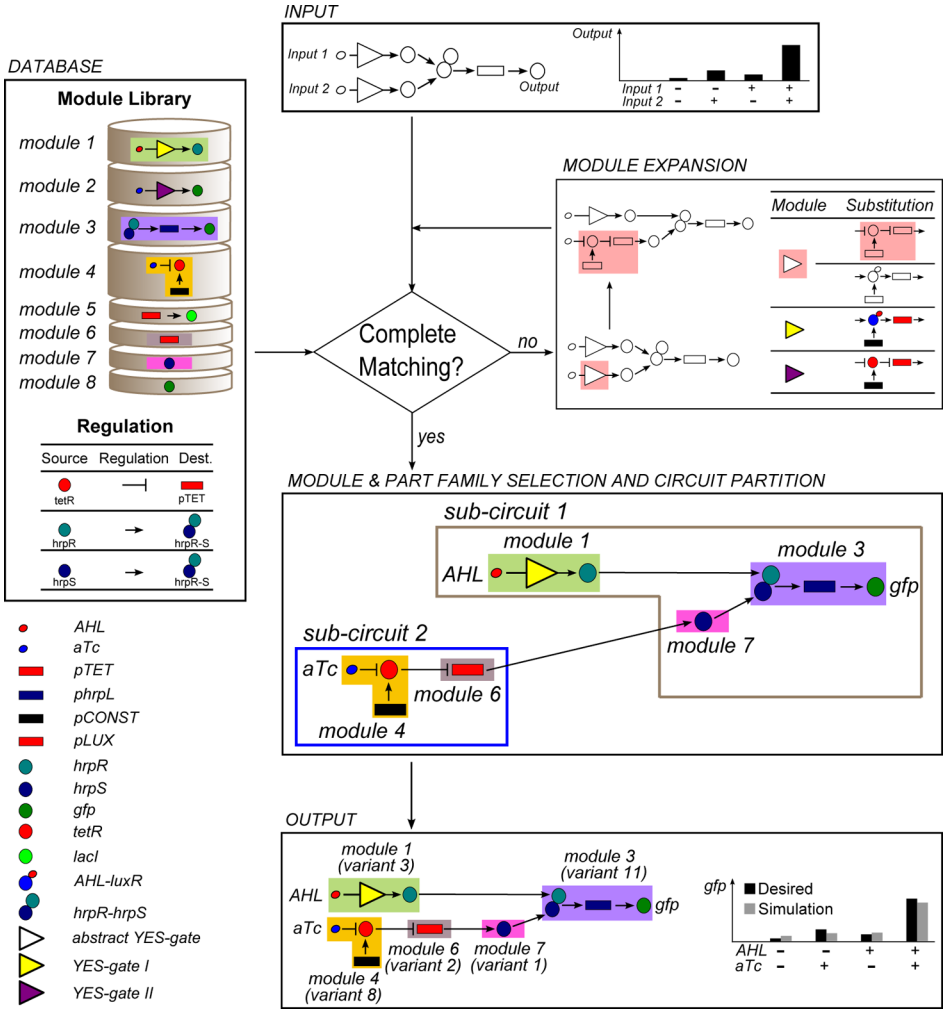


Figure 3. Example of the SBROME workflow. The user provides specifications that include the desired I/O behavior, an abstract topology, and other related constraints. SBROME searches its module database for possible matches between modules and subgraphs in the user-defined abstract network. If the design cannot be fully specified as given, the unmatched objects are expanded through a module expansion step, where objects with higher level of abstraction are substituted by topologies that contain only fundamental parts (here the YES gate is expanded to one of the two known topologies). This expansion continues until the circuit has been fully specified. In the next step, SBROME partitions the design in loosely connected subcircuits and proceeds to the final step, which is the mutant/variant selection. In that step, a specific variant of the defined part is selected from a mutant library (e.g., TetR promoters with mutations that alter downstream expression) so that the whole circuit behavior approximates the one specified by the user.

4. $V_O \subseteq V$ is a set of output nodes,

5. $\tau_1: V \rightarrow T_V$ is the node type function that assigns each node $v \in V$ with a node type $\tau_1(v) \in T_V$,

6. $\tau_2: E \rightarrow T_E$ is the edge type function that assigns each edge $e \in E$ with an edge type $\tau_2(e) \in T_E$,

7. $\nu: V \rightarrow L_V$ is the label function that assigns each node $v \in V$ with a node name $\nu(v)$.

Let $\mathcal{G}(T_V, L_V, T_E)$ denote the set of all circuit graphs for given sets T_V, L_V, T_E.

A circuit may contain nodes with name values set to "unknown" to support an abstract topology, which is formally defined as follows.

Definition 2. For given sets T_V, L_V, T_E. Each element in the set $\mathcal{H}(T_V, L_V, T_E) = \{(V, E, V_I, V_O, \tau_1, \tau_2, \nu) \in \mathcal{G}(T_V, L_V \cup \{unknown\}, T_E) | \exists v \in V: \nu(v) = unknown\}$ is called an abstract circuit topology.

As shown in Table 1, the node type set T_V contains six molecular species, and the edge type set T_E contains two relationship types. In this basic setting, however, the user is restricted to use an explicitly defined circuit topology, which is limiting as in many cases only the functional behavior is known and not the specific topological design. For this reason, SBROME groups specific subcircuits that carry out the same function into one group that is represented with a single node in the graph. By doing that, SBROME captures the *logical and/or functional behavior* of that node and proceeds to determine the actual physical design later on. To accommodate this, the node type set T_V is extended by adding five more *functional node* types (Table 1). Since T_{V_M} and T_{V_F} denote the set of molecular species and functional node types, respectively, the complete node type set is $T_V = T_{V_M} \cup T_{V_F}$.

SBROME uses two node mapping functions to substitute and expand abstract functional nodes. The first is a topological substitution of an abstract functional node with a specific physical topology. This function is performed when a match has been identified between a functional node and a specific module in the database. Its format definition is the following:

Definition 3. A substitution is a mapping $\alpha: T_{V_M} \times L_V \rightarrow \mathcal{G}(T_V, L_V, T_E)$ that maps each functional node of type $t \in T_{V_M}$ and name $l \in L_V$ to a subcircuit $\alpha(t, l) \in \mathcal{G}(T_V, L_V, T_E)$.

dx.doi.org/10.1021/sb300095m | ACS Synth. Biol. 2013, 2, 263−273

The second function is the topological expansion of an abstract functional node to a physical topology with elements that are not yet specified. The *expansion* function is used when a match between a functional node and any module in the database cannot be found and hence the *substitution* function cannot be applied. The expansion of the functional node to a physical topology allows the platform to search for possible mappings for each of the corresponding expanded parts (nodes). The *expansion* function is defined as follows.

Definition 4. A module expansion is a mapping β: $T_{V_M} \rightarrow 2^{\mathcal{H}(T_V, L_V, T_E)}$ that maps each functional node of type $t \in T_{V_M}$ to a set of abstract topology circuits $\beta(t) = \{H_1, H_2, ..., H_k\}$ where $H_i \in \mathcal{H}(T_V, L_V, T_E)$ for all $i = 1, ..., k$.

Module Query. To reuse existing modules, the module database, which contains circuits that belong to the $\mathcal{G}(T_V, L_V, T_E)$ set, is queried for matches against the transformed circuit which is an element of the $\mathcal{H}(T_V, L_V, T_E)$ set. The procedure ModuleMatching that is described in algorithm 1 is used for the purpose of finding isomorphic (sub)graphs. This procedure uses a greedy search algorithm with respect to module size and confidence level of circuit reliability, if this information is available. Generally, large modules that have been marked as validated have higher priority in the matching algorithm. Modules in the module library L are sorted by their priority order and are matched one by one with the circuit graph through the method Match. During this procedure, the method IsComplete checks if all nodes of the abstract topology have been specified. Once a matching module has been identified, the procedure Update creates a copy G' of the abstract circuit G, in which the matched nodes in G are replaced by the corresponding nodes in the matching module. In the case where one or more matched nodes are functional nodes that are still abstract (e.g., an AND gate without a specific topology), their physical topology is specified through the substitution process α. A focal point of the procedure ModuleMatching is the method Match that finds all matches between a module and an abstract circuit topology. For circuit topologies that are less than a few hundred nodes and in the presence of unspecified nodes where no other assumptions can be taken into account, the backtracking graph isomorphism algorithm proposed by Ullmann[28] has the highest performance (see review in ref 29), and this is what we have adopted here.

The ModuleMatching procedure is part of the TopologyDerivation procedure that aims to derive k possible specified circuit topologies from a given abstract topology, and it is described in algorithm 2. In the case where the ModuleMatching procedure fails to produce k possible fixed topologies, due to the lack of matching modules, a functional node from the abstract topology is chosen and then expanded to an abstract physical topology by calling the procedures ExtractFunctionComponent and ExtractImplementation, respectively. This is achieved through the expansion function β (definition 4) to retrieve a list of possible topologies that correspond to the same functional behavior. It is important to clarify that this procedure will lead to an abstract physical topology (as is depicted for the expansion of the abstract YES gate in Figure 3). This will allow the nodes within the expanded topology to be further matched by the TopologyDerivation procedure. The procedure TopologyDerivation continues recursively until it has found k fixed topologies (with k being a parameter that is defined by the user) or has determined that a module and part matching solution does not exist. Currently, there is no cost function for evaluating the quality of a match. Once a match has been identified, the respective subcircuit in the original graph is contracted into one node (i.e., module overlap is not allowed), and the search continues until all modules have been queried. The final circuit graph is then partitioned to smaller subcircuits so mutant search can be applied efficiently (Figure 3).

Algorithm 1: Module matching

```
Input: Abstract circuit topology graph G, module library L, number of solutions k
Output: List of fixed circuits (Solutions)
begin
    if (k ≤ 0) then
        return ∅
    if (IsComplete(G)) then
        return {G}
    else
        Solutions ← ∅
        for i ← 1 to |L| do
            Θ ← Match(G, L[i])
            if (Θ ≠ ∅) then
                for θ ∈ Θ do
                    G′ ← Update(G, θ)
                    L′ ← L[i + 1...|L|]
                    Solutions ← Solutions ∪ ModuleMatching(G′, L′, k − |Solutions|)
        return Solutions
```

Algorithm 2: Topology derivation

```
Input: Abstract circuit topology graph G, module library L, component
       implemetation set R, number of solutions k
Output: List of fixed circuits (Solutions)
begin
    Solutions ← ModuleMatching(G, L, k)
    if (|Solutions| < k) then
        Ω ← ExtractFunctionComponent(G)
        for ω ∈ Ω do
            Ψ ← ExtractImplementation(R, ω, G)
            for ψ ∈ Ψ do
                G′ ← Derive(G, ψ)
                if (¬IsTooLarge(G′)) then
                    Solutions ←
                    Solutions ∪ TopologyDerivation(G′, L, R, k − |Solutions|)
```

Circuit Partitioning. After the module search, the equivalent graph is partitioned into subgraphs so that the number of intercircuit links (i.e., links between two subcircuits) is minimized. By minimizing the intercircuit links, we identify connected components that have few dependencies to each other, which is desired because of both retroactivity among modules and computational efficiency, since the computational complexity of the mutant search increases exponentially with the number of the intercircuit links (see Complexity Analysis section). The min-cut-max-flow approach can be used here to partition the graph into two parts[30] and then continue partitioning these parts recursively until they all reach the desired size for applying mutant search (4−6 genes per subgraph). However, this method will yield only a locally optimal partition as iterative pairwise partitioning does not guarantee to produce the globally optimal partition set. Community detection or modularity search algorithms[31] can also be applied, but the resulting subcircuits may have large variation in sizes, which is undesirable as it may lead to load-balancing and feasibility issues. Instead, we use a multilevel partition algorithm[32] that can partition a graph with n nodes into k nearly equal-size parts with $O(n^4 \log n)$ complexity. Since the mixed interger nonlinear programming (MINLP) approach that we presented in ref 27 can effectively find the optimal part set for circuits of 6 genes or less, we set $k = n/6$. The algorithm guarantees subcircuit orthogonality by disallowing cases of cross-talk and collision on part outputs, where this can affect circuit performance.

Mutant Search. The final step in the SBROME framework is finding the optimal part variants. SBROME supports the existence of parts variants with distinct characteristics that can be used interchangeably to achieve a better match to the desired system dynamics. For instance, we have recently developed and experimentally tested a library of pLAC promoters through random

mutagenesis, each of which corresponds to different levels of downstream expression. Algorithm 3 gives in pseudocode how this "mutant" or variant search is performed. First we need to discretize the values of intercircuit links since they have continuous values and otherwise the optimization problem would be intractable. The number of discrete values is determined by the "resolution" parameter R that is user-defined. Once this is set, we use the mixed-integer nonlinear programming technique that was presented in ref 27 to find the optimal variants within each mutant library. As an alternative method, SBROME uses a genetic algorithm approach that has been found to provide similar approximate solutions to the link discretization method but without the ability to provide a bounded solution. The nonlinear expression model that is presented in the next section is used in both methods (i.e., the heuristic genetic algorithm and exact MINLP method) to describe the regulatory effect of transcription factors to protein expression and drive the MutantSearch procedure in algorithm 3. The total error of the final solution is calculated as the sum of individual subcircuit approximation errors for all subcircuits. In the future, stochastic microbial simulators[33−35] and whole-cell models[36] may also be integrated to simulate the top ranked circuits and drive further their refinement.

Algorithm 3: Mutant Search

Input: Mutant library Lib, circuit C, sub-circuit set S, linkage set $L = \{l_1, \ldots, l_k\}$
Signal inputs I, signal outputs O, resolution $R = \{r_{l_1}, \ldots, r_{l_k}\}$
Output: Optimal mutant set M
begin
  $(v_L^{min}, v_L^{max}) \leftarrow$ `EstimateBound`(C, I, Lib)
  $// v_L^{min} = (v_{l_1}^{min}, \ldots, v_{l_k}^{min})$, $v_L^{max} = (v_{l_1}^{max}, \ldots, v_{l_k}^{max})$
  for $l_i \in L$ **do**
    $\Delta_{l_i} \leftarrow v_{l_i}^{max} - v_{l_i}^{min}$
    $D_{l_i} \leftarrow \{v_{l_i}^{min}, v_{l_i}^{min} + \frac{\Delta_{l_i}}{r_{l_i}}, \ldots, v_{l_i}^{max}\}$
  for $v \in D_{l_1} \times \ldots \times D_{l_k}$ **do**
    for $s \in S$ **do**
      $(I_s, O_s) \leftarrow$ `Extract`(v, I, O)
      `Optimize`(s, Lib, I_s, O_s)
    `UpdateBestMutant`(M)

Gene Expression Model. To estimate gene expression, SBROME uses the nonlinear model that has been introduced in ref 27 and incorporates regulation, degradation, transcription, and translation. The concentration rate for protein i is given by

$$\frac{\mathrm{d}f_i}{\mathrm{d}t} = \sum_{k \in pro(i)} \left(\alpha_{0k} + \alpha_k \prod_{a \in act(k)} \frac{\beta_{ak} f_a^{\eta_{ak}}}{1 + \beta_{ak} f_a^{\eta_{ak}}} \times \prod_{r \in rep(k)} \frac{1}{1 + \beta_{rk} f_r^{\eta_{rk}}} \right) - (d_i + \mu) f_i \tag{1}$$

where $f_i(t), f_a(t)$, and $f_r(t)$ are the concentration at time point t of proteins i, a, and r, respectively, and $pro(i)$ is the set of all promoters that are upstream of the one or more copies of gene i. The various promoters may include transcription factor binding sites (TFBS) that will be part of the *cis*-regulatory region of a gene. For each promoter k in $pro(i)$ the (possibly empty) sets $act(k)$ and $rep(k)$ contain all activator and repressor proteins that are present in promoter k, respectively. For each promoter k in $pro(i)$, α_{0k} and α_k are its basal production and protein synthesis coefficient, η_{ak} and η_{rk} are the cooperativity coefficients for activator a and repressor r, and β_{ak} and β_{rk} are the binding affinities of activator a and repressor r. The degradation of protein i is captured by parameter d_i. The growth rate is represented with μ, and it is considered to be zero in stationary phase.

In many cases, gene expression is controlled by exogenously applied chemicals that induce gene expression through molecular binding. We can incorporate the effect of ligands by explicitly modeling the total amount of any protein j in the cell as the sum of the free (f_j^{free}) and ligand-bound protein (f_j^{bound}), which results in the following Hill equation model:

$$f_j = f_j^{\text{free}} + f_j^{\text{bound}} \tag{2}$$

$$f_j^{\text{free}} = \frac{\theta^\eta f_j}{\theta^\eta + [\text{ligand}]^\eta} \tag{3}$$

$$f_j^{\text{bound}} = \frac{[\text{ligand}]^\eta f_j}{\theta^\eta + [\text{ligand}]^\eta} \tag{4}$$

where [ligand] is the ligand concentration, η is the Hill coefficient (cooperativity factor), and θ is the dissociation constant. Note that eqs 2−4 apply for both activators and repressors and in cases where binding of the ligand renders the transcription factor either active or inactive. For example, when ligand binding to the transcription factor activates transcription (as it is the case with araC and L-arabinose), then the activator concentration f_a in the RHS of eq 1 is given by f_a^{bound} from eq 4. All necessary parameters for calculating the expression levels for each molecular species that participate in the circuit design are provided by the module and parts databases.

Complexity Analysis. The worst-case complexity of the module matching procedure is $O(2^n m^k)$ where m is the circuit size (i.e., number of nodes in the graph representation respectively), n is the number of modules in the library, and k is the maximum module size (i.e., the maximum number of nodes in each module graph). Although this worst-case complexity is exponential, the running time in pratice is much better since we take advantage of the look-ahead information to prune a lot of unnecessary cases through the Ullman algorithm[28] and through the greedy algorithm.

For the mutant search, suppose that we have n genes and k promoter mutants to select from, for every gene. With exhaustive search, we need to search all k^n possible combinations. In our approach, if we partition the circuit into d modules and each module has 2θ "linkage" edges on average, each represented by l expression levels, we need at most $O(n^4 \log n)$ to partition the circuit graph. In addition, searching for all possible combinations of linkage protein concentrations yields a $O(l^{\theta d} d k^{n/d})$ complexity. Therefore, the total computational complexity is $O(n^4 \log n) + O(l^{\theta d} d k^{n/d})$, which is less than that of the exhaustive search approach when $n \log k > d(\theta d \log l + \log d)/(d-1)$. The speed up will greatly increase with library expansion (i.e., higher k) or circuit complexity (i.e., higher n). The downside of the proposed method is that this is achieved at the expense of global optimality guarantee, since we have to impose discrete concentration levels for the linkage edges. Still, since we perform global optimization at the module level and propose a scheme to reuse past modules for future designs, this approach has the potential to be used through automatic circuit design of very large number of components.

RESULTS

Multiplexer Circuit. To evaluate the capacity of the SBROME framework to yield synthetic circuits with the desired characteristics, we assessed its performance with the multiplexer circuit. The choice of the multiplexer circuit as a proof-of-concept and evaluation case was based on the following criteria: (a) a synthetic circuit with adequate complexity so it can be decomposed into smaller subcircuits, (b) a circuit that can be designed so that it reuses other synthetic circuits that have been experimentally

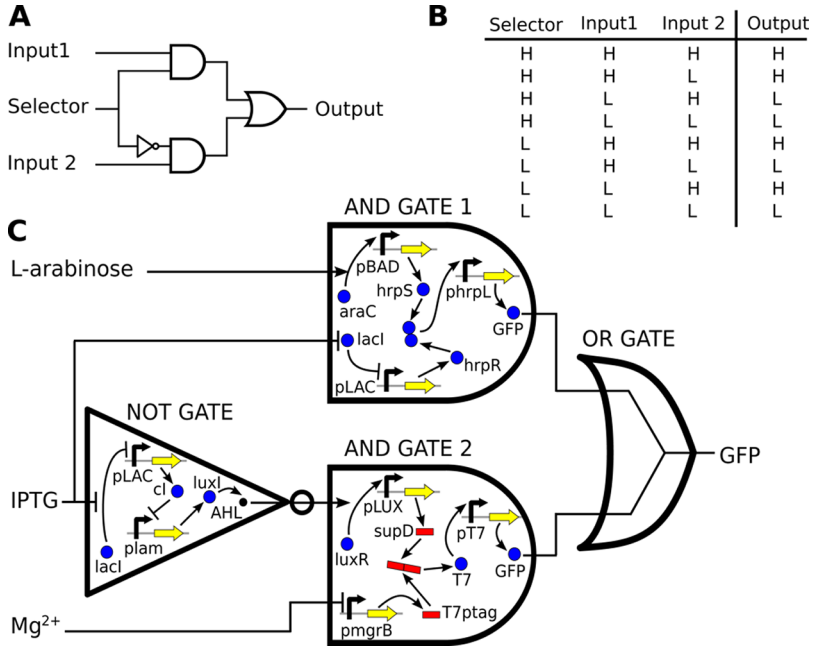


Figure 4. Implementation of a multiplexer circuit. (A) A design with two AND gates, one NOT gate, and one OR gate. (B) Truth table for the multiplexer circuit. (C) A physical implementation of the multiplexer design shown in panel A.

verified, and (c) a circuit that exhibits an analog behavior but can be characterized in its general form with a truth table. A multiplexer is a common component in electrical engineering that uses a selector signal to make the selection between two other input signals. The digital behavior of a multiplexer is illustrated by the truth table in Figure 4B: when the selector input is high, then the output is equal to the first input, whereas if the selector input is low, the second input is propagated to the output. There are several synthetic designs to make a biological multiplexer, including the designs described in refs 37 and 38. Nevertheless, the high complexity of the first circuit and the absence of a complete experimentally validated circuit for the second circuit necessitate a further investigation on how to achieve multiplexer implementations that are simpler to construct and have a considerable chance of successful operation. Toward this goal, we here use SBROME to design a multiplexer by using a standard multiplexer schematic from electrical engineering (Figure 4A with a truth table depicted in Figure 4B) and reusing previously constructed and well-characterized modules. We used a module database of 43 circuits that have been published in the past and a parts database that consists of 101 parts that have been experimentally characterized. After the application of our pipeline, our framework came up with the multiplexer design that is shown in Figure 4C that consists of a RNA−RNA AND gate (published in ref 4) and the protein−protein interaction-based AND gate from ref 5 and a NOT gate in ref 5 where the coding region for gfp is replaced by the coding region for luxI. In Figure 5, a propotype graphical user interface for SBROME is shown that includes the resulting automatic layout of the multiplexer circuit. The output expression levels for this automatically designed, modular multiplexer circuit are depicted in Figure 6. At the same time, we optimized *ab initio* our design by performing a mutant search that selects the optimal mutants for the four constitutive promoters and two pLAC promoters to achieve a given behavior. As discussed above, we used the mutant library for the constitutive promoter from ref 39 and the mutant library for the pLAC promoter from ref 40. A comparison between the solution provide by the SBROME framework and an exhaustive search is shown in Figure 6, and Table 2 depicts the running time and error of both approaches. Remarkably, the modular design exhibited a digital on/off characteristic that closely follows the truth table that was depicted in Figure 4B. In contrast, when exhaustive search was applied, the resulting circuit exhibited a more analog behavior that tracked better (hence the lower error in Table 2) the noisy "desired" signal but propagated a distorted input signal to the circuit output. Indeed, when the output concentration is compared with a "high/low" signal (Table 2), the modular design performs more closely to a binary selector.

Interestingly, if we change the topology of the circuit to an alternative design with two NOR gates, one AND gate, and one OR gate as in Figure 7A, the tool will come up with a solution where we can take advantage of the design of two NOR gates in ref 6 and the AND gate design in ref 5. For compatibility purposes the yfp gene was replaced by our framework in the second NOR gate to gfp to make the final OR wiring available. Figure 7B demonstrate this alternative multiplexer design and corresponding parts.

■ DISCUSSION

The automated circuit design methodology SBROME provides a useful framework to create modular designs that reuse pre-existing constructs. The proof-of-concept multiplexer design illustrates an additional advantage of using pre-existing modules in digital biosystems design: while global optimization of part selection is prone to individual noise levels and approximation errors, utilizing digital modules with saturating input/output functions can restore or regenerate the signal propagation and improve the input/output characteristic function of the whole circuit.

There are several extensions that will be explored for further improvement of the algorithmic and practical efficiency of SBROME. When it comes to module search, feasibility rules may be applied if we are dealing with larger graphs to speedup the search,[41] and for large module libraries, a library reorganization as in ref 42 can

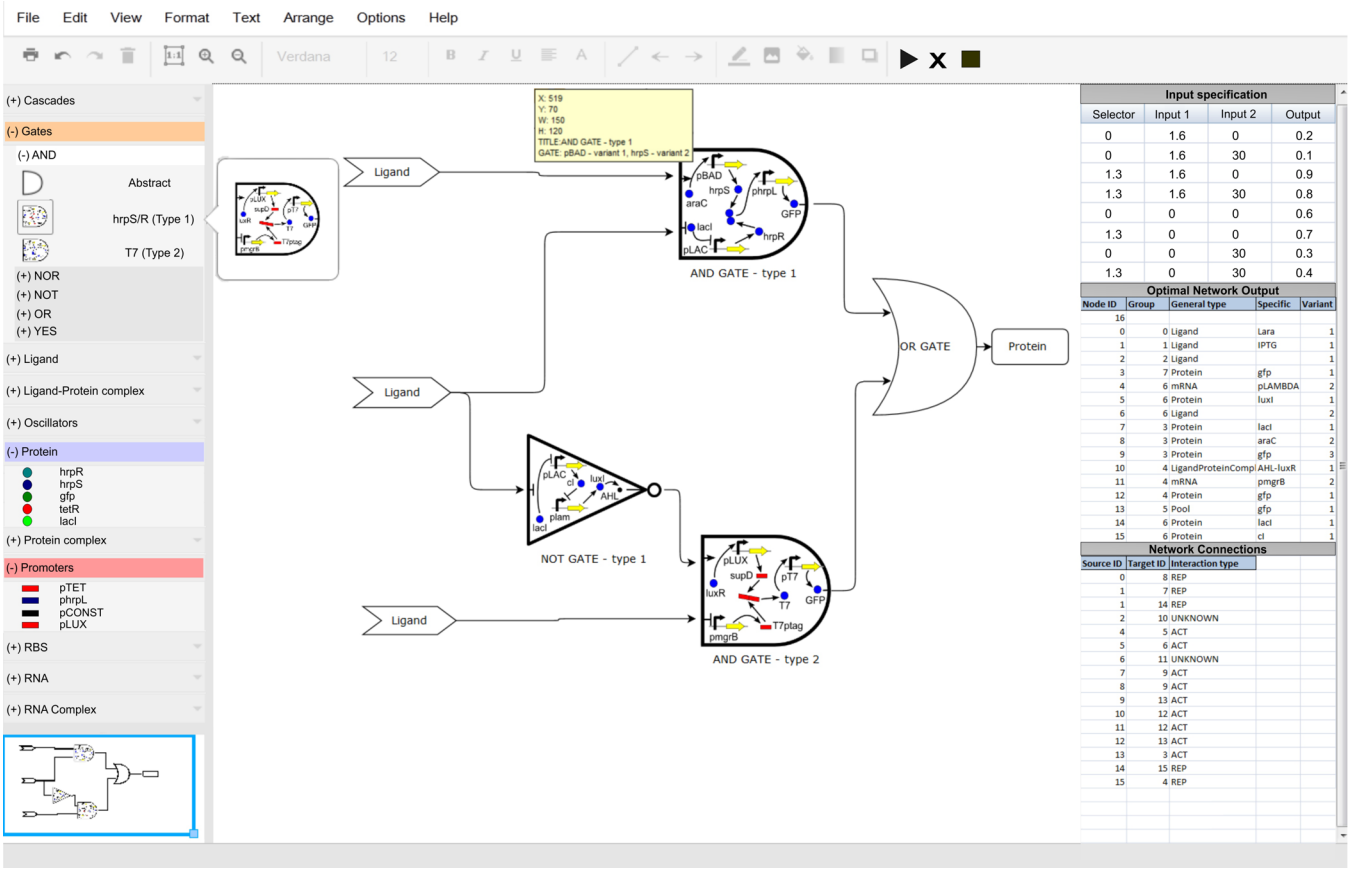


Figure 5. Graphical user interface of the SBROME platform that depicts the automatically designed multiplexer topology that best approximates the I/O characteristic that was supplied by the user.

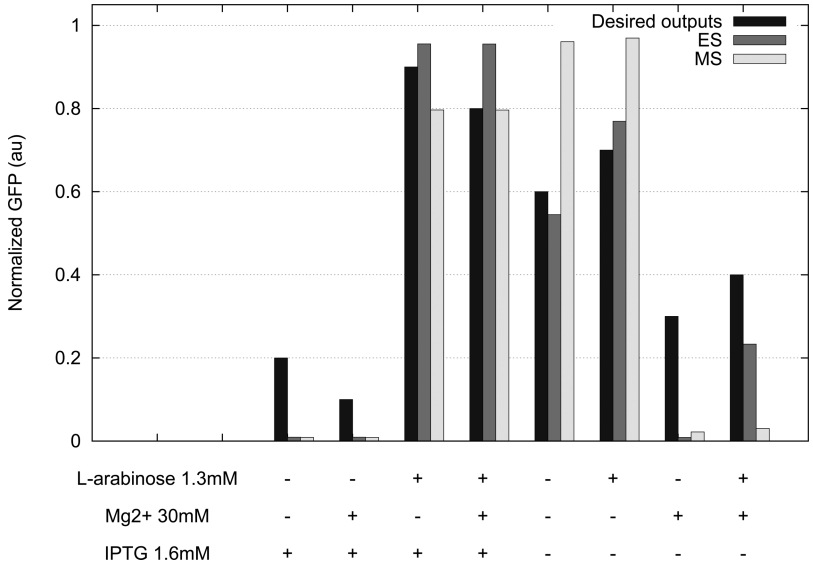


Figure 6. Comparison between solutions of modular search (MS) and exhaustive search (ES) for the multiplexer case study.

Table 2. Comparison of Steady State Running Time

		exhaustive search			modular search		
design	library size	time (s)	error	error w.r.t. digital on/off	time (s)	error	error w.r.t. digital on/off
multiplexer	$8^2 6^4$	4.5×10^3	0.19	0.32	4.2×10^2	0.47	0.09

significantly reduce the running time. Heuristic algorithms have been successfully employed in electrical circuit design to address similar challenges in module searching,[43] and similar extensions can be used here to reduce the search space and hence improve

dx.doi.org/10.1021/sb300095m | ACS Synth. Biol. 2013, 2, 263−273

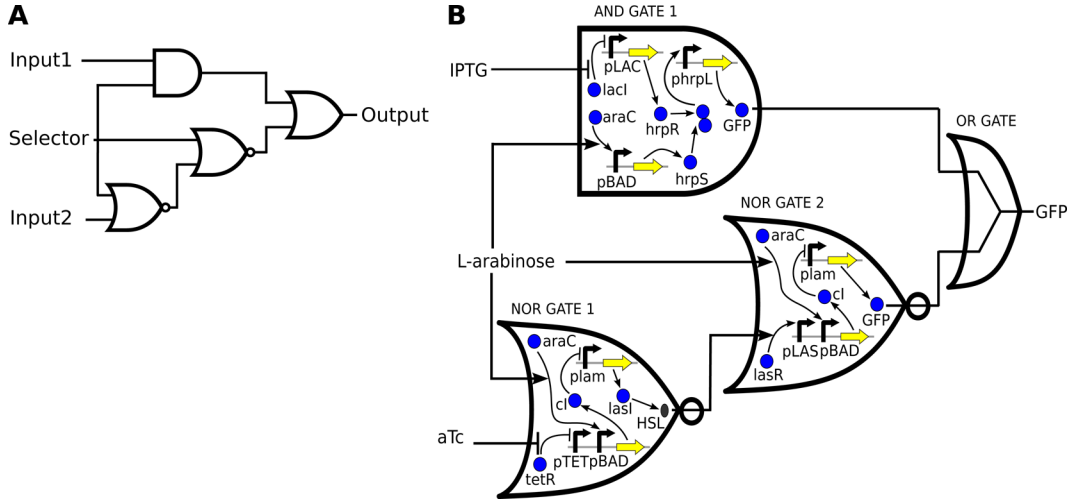


Figure 7. Alternative implementation of the multiplexer circuit. (A) A design with one AND gate, one OR gate, and two NOR gates. (B) A physical implementation of the multiplexer design shown in panel A.

computational performance. Similarly, in the mutant search step, we can introduce approximation algorithms such as approximate dynamic programing and other optimization techniques to find approximate solutions with guaranteed bounds, but in less computational time. As more biological information becomes available, the SBROME pipeline will be extended to incorporate features that can lead to more robust and reconfigurable circuits. Some of these features are the compatibility between the various modules and parts, as well as the expected cross-talk or retroactivity that may interfere with the final circuit behavior. Retroactivity, the phenomenon where a downstream module affects the behavior of an upstream component, can be detrimental for modular design as the final circuit behavior may vary considerably.[44−46] Since SBROME is a part-matching and optimization framework when the abstract design is already given, it does not directly address this challenge. Still, since the current framework minimizes the interconnections among the various subcircuits in the partition phase, it actually creates designs that tend to have low retroactivity *a priori*, if no other information is known.

Recently, a similar methodology was proposed to deal with modular design of synthetic circuits (named "MatchMaker"[24]). MatchMaker uses a two-step approach where the abstract gene regulatory network (AGRN) is first matched with a feature GRN, which is a supergraph that contains all known regulatory interactions to determine specific parts in the circuit (which is similar to phase 2 of the three-phase framework described in ref 26). In the second step, the resulting GRN (that now has "fixed" parts) is matched with a module library to figure out the optimal way to assemble the circuit from modules in this library. This innovative approach has the advantage that it can scale well with the increase in module library size, assuming that the feature GRN does not grow significantly (e.g., the increase in the engineered or native associations that are introduced is small), and the feature graph provides a useful map of all possible interactions in the parts that we have at hand. However, this approach becomes inefficient in cases where one or more matches exist between the abstract and feature GRNs, but no actual modules exist to be used for further assembly. For example, there are many ways to design a cascade with three genes, but there is only one reliable design that has been characterized experimentally,[47] in which case using a feature GRN is inefficient. In the approach presented here, we only use one step to match directly from the abstract topology to the existing module set, while at the same time we skip all cases that lead to utilizing modules that are not available in the database. This is achieved at the expense of having to ensure that the interactions between modules are preserved. Database reorganization techniques similar to the ones presented in ref 42 have the potential to decrease the module search time in any module matching method mentioned above.

Future extensions of the SBROME framework will include the incorporation of other regulatory, proteomic, and metabolic relationships. Especially in the case of the metabolic interactions, the integration of a flux balance model that is based on linear optimization is a natural extension of the part selection algorithms that are based in mathematical optimization such as the one that we introduced for mutant search. Extending the tool to take into account interactions across multiple "omics" layers will empower the user to seek optimized universal solutions and will be a step closer to an ultimate platform in automated biosystems design.

SBROME is available as a resource and online tool at http://tagkopouloslab.ucdavis.edu/software.html.

AUTHOR INFORMATION

Corresponding Author

*E-mail: itagkopoulos@ucdavis.edu.

Author Contributions

L.H. wrote the code and performed the experiments. A.T. implemented the SBROME GUI and online service. M.K. advised on the mathematical techniques. I.T. conceived the project and supervised all development and analysis. L.H. and I.T. wrote the paper.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We would like to acknowledge the support from NSF (grants 0941360 and 1146926). We thank Vadim Mozhayskiy for his help with the production of Figure 3.

REFERENCES

(1) Purnick, P. E. M., and Weiss, R. (2009) The second wave of synthetic biology: from modules to systems. *Nat. Rev. Mol. Cell Biol. 10*, 410−422.

(2) Cooling, M. T., Rouilly, V., Misirli, G., Lawson, J. R., Yu, T., Hallinan, J., and Wipat, A. (2010) Standard virtual biological parts: a repository of modular modeling components for synthetic biology. *Bioinformatics 26*, 925−931.
(3) Landrain, T. E., Carrera, J., Kirov, B., Rodrigo, G., and Jaramillo, A. (2009) Modular model-based design for heterologous bioproduction in bacteria. *Curr. Opin. Biotechnol. 20*, 272−279.
(4) Anderson, J. C., Voigt, C. A., and Arkin, A. P. (2007) Environmental signal integration by a modular AND gate. *Mol. Syst. Biol.*, DOI: 10.1038/msb4100173.
(5) Wang, B., Kitney, R. I., Joly, N., and Buck, M. (2011) Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nat. Commun. 2*, 508+.
(6) Tamsir, A., Tabor, J. J., and Voigt, C. A. (2011) Robust multicellular computing using genetically encoded NOR gates and chemical "wires. *Nature 469*, 212−215.
(7) Lou, C. (2010) Synthesizing a novel genetic sequential logic circuit: a push-on push-off switch. *Mol. Syst. Biol.*, DOI: 10.1038/msb.2010.2.
(8) Miller, M., Hafner, M., Sontag, E., Davidsohn, N., Subramanian, S., Purnick, P. E. M., Lauffenburger, D., and Weiss, R. (2012) Modular design of artificial tissue homeostasis: robust control through synthetic cellular heterogeneity. *PLoS Comput. Biol. 8*, No. e1002579.
(9) Gardner, T. S., Cantor, C. R., and Collins, J. J. (2000) Construction of a genetic toggle switch in Escherichia coli. *Nature 403*, 339−342.
(10) Litcofsky, K., Afeyan, R., R., K., Khalil, A., and Collins, J. (2012) Iterative plug-and-play methodology for constructing and modifying synthetic gene networks. *Nat. Methods 9*, 1077−1080.
(11) Ginkel, M., Kremling, A., Nutsch, T., Rehner, R., and Gilles, E. D. (2003) Modular modeling of cellular systems with ProMoT/Diva. *Bioinformatics 19*, 1169−1176.
(12) Smith, L. P., Bergmann, F. T., Chandran, D., and Sauro, H. M. (2009) Antimony: a modular model definition language. *Bioinformatics 25*, 2452−2454.
(13) Chandran, D., and Sauro, H. M. (2012) Hierarchical modeling for synthetic biology. *ACS Synth. Biol. 1*, 353−364.
(14) Pedersen, M., and Phillips, A. (2009) Towards programming languages for genetic engineering of living cells. *J. R. Soc., Interface 6*, S437−S450.
(15) Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J. C., and Densmore, D. (2011) Eugene - a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS ONE 6*, No. e18882.
(16) Goler, J. A. (2004) *BioJADE: a design and simulation tool for synthetic biological systems*, AI Technical Report 2004-003, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge.
(17) 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.
(18) 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.
(19) Chandran, D., Bergmann, F., and Sauro, H. (2009) TinkerCell: modular CAD tool for synthetic biology. *J. Biol. Eng 3*, 19.
(20) Marchisio, M. A., and Stelling, J. (2011) Automatic design of digital synthetic gene circuits. *PLoS Comput. Biol. 7*, e1001083.
(21) Rodrigo, G., Carrera, J., and Jaramillo, A. (2007) Genetdes: automatic design of transcriptional networks. *Bioinformatics 23*, 1857−1858.
(22) Wu, C.-H., Lee, H.-C., and Chen, B.-S. (2011) Robust synthetic gene network design via library-based search method. *Bioinformatics 27*, 2700−2706.
(23) Rodrigo, G., and Jaramillo, A. (2013) AutoBioCAD: full biodesign automation of genetic circuits. *ACS Synth. Biol. 1*, 1−2.
(24) Yaman, F., Bhatia, S., Adler, A., Densmore, D., and Beal, J. (2012) Automated selection of synthetic biology parts for genetic regulatory networks. *ACS Synth. Biol. 1*, 332−344.
(25) Miyamoto, T., Razavi, S., DeRose, R., and Inoue, T. (2012) Synthesizing biomolecule-based Boolean logic gates. *ACS Synth. Biol. 2*, 72−82.
(26) Hunyh, L., and Tagkopoulos, I. (2012) A robust, library-based, optimization-driven method for automatic gene circuit design, in *Computational Advances in Bio and Medical Sciences, 2012 IEEE 2nd International Conference*, pp 1−6, IEEE, New York.
(27) 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*, No. e35529.
(28) Ullmann, J. R. (1976) An Algorithm for Subgraph Isomorphism. *J. Assoc. Comput. Mach. 23*, 31−42.
(29) Messmer, B. (1995) Efficient graph matching algorithms for preprocessed model graphs. Ph.D. thesis, University of Bern, Switzerland.
(30) Hao, J., and Orlin, J. B. (1994) A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms 17*, 424−446.
(31) Newman, M. E. J. (2004) Fast algorithm for detecting community structure in networks. *Phys. Rev. E 69*, 066133+.
(32) Karypis, G., and Kumar, V. (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput. 20*, 359−392.
(33) Tagkopoulos, I., Liu, Y. C., and Tavazoie, S. (2008) Predictive behavior within microbial genetic networks. *Science 320*, 1313−1317.
(34) Mozhayskiy, V., Miller, B., Ma, K.-L., and Tagkopoulos, I. (2011) A scalable multi-scale framework for parallel simulation and visualization of microbial evolution,in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, 7:1−7:8, Association for Ccomputing Machinery, New York.
(35) Mozhayskiy, V., and Tagkopoulos, I. (2012) Horizontal gene transfer dynamics and distribution of fitness effects during microbial in silico evolution. *BMC Bioinformatics 13*, S13.
(36) Karr, J. R., Sanghvi, J. C., Macklin, D. N., Gutschow, M. V., Jacobs, J. M., Bolival, B., Assad-Garcia, N., Glass, J. I., and Covert, M. W. (2012) A whole-cell computational model predicts phenotype from genotype. *Cell 150*, 389−401.
(37) Moon, T. S., Clarke, E. J., Groban, E. S., Tamsir, A., Clark, R. M., Eames, M., Kortemme, T., and Voigt, C. A. (2011) Construction of a genetic multiplexer to toggle between chemosensory pathways in Escherichia coli. *J. Mol. Biol. 406*, 215−227.
(38) Pasotti, L., Quattrocelli, M., Galli, D., De Angelis, M. G. C., and Magni, P. (2011) Multiplexing and demultiplexing logic functions for computing signal processing tasks in synthetic biology. *Biotechnol. J. 6*, 784−795.
(39) Berkeley 2006 iGEM Team. Constitutive promoter family (retrieved August, 2012). http://partsregistry.org/Promoters/Catalog/Anderson.
(40) Ellis, T., Wang, X., and Collins, J. (2009) Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat. Biotechnol. 27*, 465−471.
(41) Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. (2004) A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 1367−1372.
(42) Messmer, B. T., and Bunke, H. (2000) Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Trans. Knowl. Data Eng. 12*, 307−323.
(43) Ohlrich, M., Ebeling, C., Ginting, E., and Sather, L. (1993) SubGemini: identifying subcircuits using a fast subgraph isomorphism algorithm, in *Proceedings of the 30th international Conference on Design Automation*, pp 31−37, IEEE, New York
(44) Del Vecchio, D., Ninfa, A. J., and Sontag, E. D. (2008) Modular cell biology: retroactivity and insulation. *Mol. Syst. Biol. 4*, 161.
(45) Saez-Rodriguez, J., Gayer, S., Ginkel, M., and Gilles, E. D. (2008) Automatic decomposition of kinetic models of signaling networks minimizing the retroactivity among modules. *Bioinformatics 24*, i213−i219.
(46) Voigt, C., Ed. (2011) *Synthetic Biology: Methods for Building and Programming Life*, Methods in Enzymology Vol. 497, Academic Press, New York.

(47) Hooshangi, S., Thiberge, S., and Weiss, R. (2005) Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. *Proc. Natl. Acad. Sci. U.S.A. 102*, 3581−3586.
(48) Guido, N. J., Wang, X., Adalsteinsson, D., Mcmillen, D., Hasty, J., Cantor, C. R., Elston, T. C., and Collins, J. J. (2006) A bottom-up approach to gene regulation. *Nature 439*, 856−60.
(49) Tabor, J. J., Salis, H. M., Simpson, Z. B., Chevalier, A. A., Levskaya, A., Marcotte, E. M., Voigt, C. A., and Ellington, A. D. (2009) A synthetic genetic edge detection program. *Cell 137*, 1272−1281.
(50) Canton, B., Labno, A., and Endy, D. (2008) Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol. 26*, 787−793.
(51) Sayut, D. J., Niu, Y., and Sun, L. (2009) Construction and enhancement of a minimal genetic AND logic gate. *Appl. Environ. Microbiol. 75*, 637−642.
(52) Elowitz, M. B., and Leibler, S. (2000) A synthetic oscillatory network of transcriptional regulators. *Nature 403*, 335−338.
(53) Stricker, J., Cookson, S., Bennett, M. R., Mather, W. H., Tsimring, L. S., and Hasty, J. (2008) A fast, robust and tunable synthetic gene oscillator. *Nature 456*, 516−519.
(54) Atkinson, M. R., Savageau, M. A., Myers, J. T., and Ninfa, A. J. (2003) Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in Escherichia coli. *Cell 113*, 597−607.