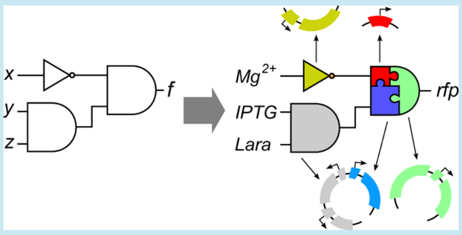
Optimal Part and Module Selection for Synthetic Gene Circuit Design Automation

Linh Huynh and Ilias Tagkopoulos*

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

Ⓢ *Supporting Information*

ABSTRACT: An integral challenge in synthetic circuit design is the selection of optimal parts to populate a given circuit topology, so that the resulting circuit behavior best approximates the desired one. In some cases, it is also possible to reuse multipart constructs or *modules* that have been already built and experimentally characterized. Efficient part and module selection algorithms are essential to systematically search the solution space, and their significance will only increase in the following years due to the projected explosion in part libraries and circuit complexity. Here, we address this problem by introducing a structured abstraction methodology and a dynamic programming-based algorithm that guaranties optimal part selection. In addition, we provide three extensions that are based on symmetry check, information look-ahead and branch-and-bound techniques, to reduce the running time and space requirements. We have evaluated the proposed methodology with a benchmark of 11 circuits, a database of 73 parts and 304 experimentally constructed modules with encouraging results. This work represents a fundamental departure from traditional heuristic-based methods for part and module selection and is a step toward maximizing efficiency in synthetic circuit design and construction.



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

Similar to any other engineering-related field, computer-aided design (CAD) tools in synthetic biology are essential for designing circuits faster, better and more reliably. Numerous computational CAD tools have been developed recently for the design and implementation of synthetic gene circuits (a recent review of these CAD tools available in ref 1).

A step toward automated synthetic circuit design is the selection of a part set so that the final circuit exhibits the desired behavior. Parts can either be *elementary*, such as a promoter or a gene, or *composite* that consist of two parts or more, hence constituting *modules*. The various approaches that have been introduced so far address successfully this problem. However, all of them have one or more of the following limitations: (i) the input circuit is specified at a low abstraction level that is construction-based (e.g., a set of promoters, coding regions, and so on) rather than at a higher abstraction level that would be function-based (e.g., truth table, logical gate netlist, custom analog functions, and so on), (ii) a higher abstraction level specification is used but the biological diversity (i.e., one functional behavior can be carried out by many different biological processes) is not considered, (iii) the recycling of past circuits or modules is not considered and (iv) there are no guaranties or bounds on the selection optimality.

Table 1 summarizes the features of all CAD tools that can support the selection of parts and modules to design a synthetic gene circuit. The GEC language[2] was the first tool that allowed the user to specify the circuit at a high abstraction level as a program and then a compiler would translate this program into sequences of genetic parts, although biological diversity, optimality, and module reuse are not considered. Another notable example is iBioSim[3] that can take into account the module reuse, although it only operates at a low specification level. In contrast, this work by Marchisio et al.[4] allows the user to specify a functional circuit behavior by a truth table and then to automatically construct a topology, albeit without module reusability or optimality. A recent work-flow called TASBE[5] goes a step further, as it allows the user to specify the circuit behavior at a language level, and then, it uses the heuristic algorithm from MatchMaker[6] to select parts and modules, although it does not take into account biological diversity and optimality considerations. More recently, we developed the SBROME framework[7] that allows circuit specification by both netlists and analog I/O functions, while it takes into account the biological diversity and module reuse. However, both the

Table 1. Feature Comparison of Various SynBio CAD Tools

tool	ref	abstraction level of circuit specification	accounts for biological diversity	module reuse supported	provides optimal part allocation
GEC	2	high	no	no	no
iBioSim	3	low	no	yes	yes
CAD tool	4	high	yes	no	no
TASBE	5	high	no	yes	no
SBROME	7	high	yes	yes	no
this work		high	yes	yes	yes

Special Issue: IWBDA 2013

Received: September 16, 2013

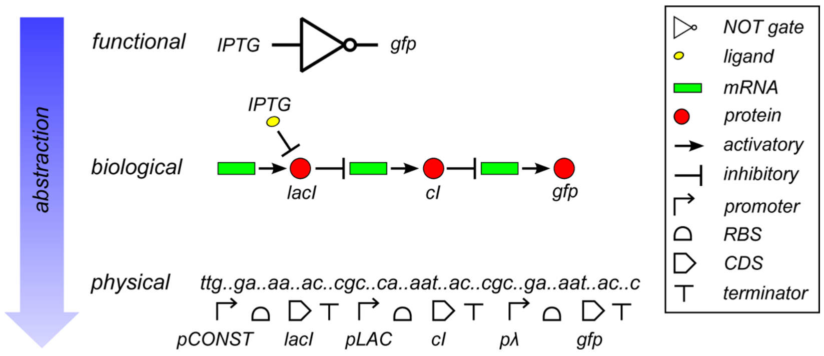


Figure 1. Overview on the circuit representation with different abstraction levels.

topology searching (i.e., finding a biological network motif for each logical gate) and the part and module selection are still partially based on heuristic algorithms that do not guarantee optimality.

There is currently no method that can identify the smallest number of parts and modules to construct a circuit with a specified function given by a high-level specification, handle the biological diversity efficiently and take into account the construction optimization by reusing modules. The integration of these characteristics in a single tool would be desireable, since abstract circuit specification and accounting for the biological diversity enlarge the solution space and design flexibility while the optimal selection of parts and modules can reduce the time and cost to produce the final circuit. Efficient part and module selection algorithms are essential to systematically search the solution space and their significance will only increase in the following years due to the projected explosion in part libraries and circuit complexity.[8]

This paper presents a method that incorporates all these characteristics. We first classify the gene circuit specification into three abstraction levels as in Figure 1. By doing so, the user can specify their design at any combination of the two highest abstraction levels (i.e., functional and biological) while parts and modules curated from literature are specified at the lowest abstraction level. To find an optimal circuit construction, we need to find a part/module set that can be assembled together to match with the user specification. Since the circuit and part/module library specifications are at different abstraction levels, our method first transforms all specifications into the second abstraction level (biological) and then performs the part selection. While the conversion for the specification of parts and modules from a lower level to an upper level is straightforward, the transformation of the user circuit specification from the highest level to the intermediate one by a naive approach (i.e., consider all possible network motif combinations) leads to a combinatorial explosion because of the biological diversity present. To avoid that, we merge all the combinations into an unique graph to exploit the overlap between these combinations. To select a set of parts and modules that optimizes the circuit construction, we define a cost function of a set of parts and modules that can approximate the number of amplification/ligation steps to build the circuit from that part/module set. For that cost function, we develop a dynamic programming (DP) based algorithm that adheres to the same principles as technology mapping that is used in digital electronic circuits,[9] but it is adapted to this application so it takes into account the cross-talk effects and the compatibility checking. These necessary modifications, however, introduce an exponential worst-case complexity, which can significantly impact the algorithm ability to provide solutions for large libraries or large circuits. To overcome this problem, we introduce three techniques to improve the computational performance of the base method, that are based on symmetry checking, information look-ahead, and branch-and-bound. After integrating the extended method into the SBROME CAD framework,[7] we evaluated our approach with a benchmark of 11 different circuits and a library of 73 parts and 304 modules that were identified from an extensive literature curation that focused on *E. coli* constructs published the past 20 years.

■ RESULTS

Part and Module Library. We curated the literature to construct a part and module library. To build the module library, we collected the description of 102 *E. coli* plasmids that were published in 19 papers over the past 2 decades. Each module is constructed from a plasmid fragment as described in the Methods section. The final library, which contains 73 parts and 304 modules, is described as in Table 2.

Table 2. Part and Module Library

library description	quantity
parts	73
promoters	31
coding regions	40
noncoding regions	2
modules	304
plasmids	102
references	19

Benchmarking and Performance. We propose the method of Motif-Merging Dynamic Programming (MMDP) that consists of two steps, which are the expansion step to expand from functional behaviors to biological processes and the selection step to select an optimal set of parts and modules. For the first step, we propose to merge all topology motif combinations into an unique graph (algorithm 1 in the Methods section) together with an extension (see the Methods section). For the second step, we propose to apply a DP-based algorithm (algorithm 2 in the Methods section) together with two extensions (see the Methods section). A benchmark that contains 11 circuits (Table 3, D1 and D2 are two circuits described in ref 10) is used to evaluate the proposed methodology. The proposed solution cost (which is the total cost of parts and modules used in this solution, see the

Table 3. Result Summary[a]

design	no. of gates	no. of possible topological motif combinations	no. of nodes (merged topology)	proposed solution cost: MMHS	proposed solution cost: MMDP, ESDP	running time (second): ESDP	running time (second): MMHS	running time (second): MMDP
2-cascade	2	3	21	$2 - \varepsilon$	$1 - \varepsilon$	2.6×10^{-1}	8.0×10^{-2}	1.3×10^{-1}
3-cascade	3	3	24	$3 - \varepsilon$	2	3.2×10^{-1}	7.0×10^{-2}	1.7×10^{-1}
4-cascade	4	3	27	$3 - \varepsilon$	$3 - \varepsilon$	4.3×10^{-1}	9.0×10^{-2}	1.8×10^{-1}
band-detector	3	6	35	6	5	3.9	2.6×10^{-1}	1.5
feed-forward	3	18	49	2	2	7.9	1.6×10^{-1}	3.1×10^{-1}
not-and	2	27	51	2	2	8.6	2.2×10^{-1}	4.5×10^{-1}
3-in-and	2	324	80	3	$3 - 3\varepsilon$	3.1×10^{2}	4.5×10^{-1}	1.2
3-in-not-and	3	432	81	$5 - \varepsilon$	$4 - 2\varepsilon$	4.6×10^{2}	4.4×10^{-1}	8.8
2-to-1-mux	4	972	94	7	$5 - 3\varepsilon$	2.8×10^{3}*	1.3	3.2×10^{1}
D1	4	11664	124	7	$6 - 4\varepsilon$	4.9×10^{4}*	8.5×10^{-1}	2.4×10^{2}
D2	5	8748	127	8	$6 - 4\varepsilon$	3.8×10^{4}*	1.9	3.7×10^{2}

[a]The first three columns depict the input topology size (i.e. the number of logic gates), the number of all topology motif combinations and the number of nodes of the merged graph. The proposed solution cost and the running time for different approaches are showed in last five columns (* refer to estimated). MMHS (Motif-Megre Heuristic-Search), MMDP (Motif-Merge Dynamic-Programming), and ESDP (Motif-Exhaustive-Search Dynamic-Programming) respectively denote the method that merges all motif topology combinations (algorithm 1) and apply a heuristic search, the method that merges all topology motif combinations (algorithm 1) and apply a DP-based algorithm (algorithm 2) with extensions, and the method that traverses all topology motif combinations and applies a DP-based algorithm (algorithm 2) with extensions for each combination.

Methods section) and the running time of our method for each circuit is presented in Table 3. For circuits that have been constructed experimentally (the 2-cascade and the 3-input AND gate), our method can return the solution that has been reported in the literature (the 2-cascade in ref 11 and the 3-input AND gate in ref 12, respectively). Interestingly, for the design D2 that was reported in ref 10 for mammalian cells, our method resulted in a simple solution for *E. coli* cells by utilizing chaperone proteins as shown in Figure 2.

To evaluate its performance, we first evaluate MMDP with a naive Exhaustive Search DP-based algorithm (ESDP) that traverses all topology motif combinations sequentially and applies our proposed DP-based algorithm and extensions for each combination to find the finally optimal set of parts and modules. Since the DP-based algorithm is an exact algorithm and the merged graph encodes all combinations, the optimal sets that are found by these two methods (i.e., MMDP and ESDP) are the same. Therefore, we only compare the running time of those methods. The comparison result that is presented in Table 3 shows that our method is faster than the naive approach and the speed up increases with the circuit size. In general, for a circuit with n gates, each gate has m topology motifs and each motif has k nodes in average then there are totally $\Theta(m^n)$ topology motif combinations while the size of the merged graph that encodes all these combinations is only $\Theta(mnk)$. Suppose that the computational complexity of the DP-based algorithm with extensions is $\varphi(x)$ for a graph with x nodes. The computational complexity of our method and the ESDP method is $\Theta(\varphi(mnk))$ and $\Theta(m^n\varphi(nk))$, respectively. Therefore, if φ is a quasi-polynomial function (i.e., it does not increase so fast as an exponential function) then our method will be faster and the speed up increases for a larger circuit size n because of the term m^n. To illustrate this analysis, the total number of topology motif combinations (i.e., $\Theta(m^n)$) and the merged graph size (i.e., $\Theta(mnk)$) of each circuit are also presented in Table 3.

Second, we compare our method with an alternative method (denoted by MMHS for Motif-Merge Heuristic-Search) that also merges all topology motif combinations into an unique graph (i.e., it also applies algorithm 1 with an extension), but then, it applies a heuristic search approach (introduced in the branch-and-bound extension in the Methods section) to select the part/module set. We compare their proposed solution cost and running time. Since both methods (i.e., MMDP and MMHS) use the same approach for the first step (i.e., algorithm 1 with an extension), this comparison is actually to compare between the DP-based algorithm with extensions and a heurisric search algorithm to select the part/module set in the second step. The comparison result that is presented in Table 3 shows that although our method is slower than the MMHS method, it proposes a lower cost (which means a more compact part/module set) than the one of the MMHS method (8 of 11 cases). This is a significant and practical improvement over heuristic based methods since the extra computation time required by our method is much less than the time needed to construct the circuit from a part/module set with a larger cost.

DISCUSSION

In this work, we described a part and module selection strategy that returns the optimal set of parts and modules for synthetic gene circuit design automation. There are several extensions of this work that warrant further investigation. First, our selection method has to store a set of all subsolutions of each node to check the contraints of the cross-talk absence and the connection compatibility. Hence, we can improve the computational performance more by reorganizing this subsolution set as a hierarchical tree instead of a linear list as in the current method (algorithm 2 in the Methods section). Second, the current method is limited to circuits that topology is a single output and directed acyclic graph (i.e., the graph contains only one output node and no directed loop). Therefore, extending this method to general graphs will be useful in cases where designs contain loops and/or multiple outputs. A solution would be to decompose the circuit into subcircuits that have a directed acyclic, single output topology. After applying the algorithms presented here, we can combine each matchings together to form a solution for the whole circuit, provided that both the cross-talk and compatibility constraints are satisfied.

More far-reaching and ambitious extensions of this method include its application in multi-cellular system design, where the

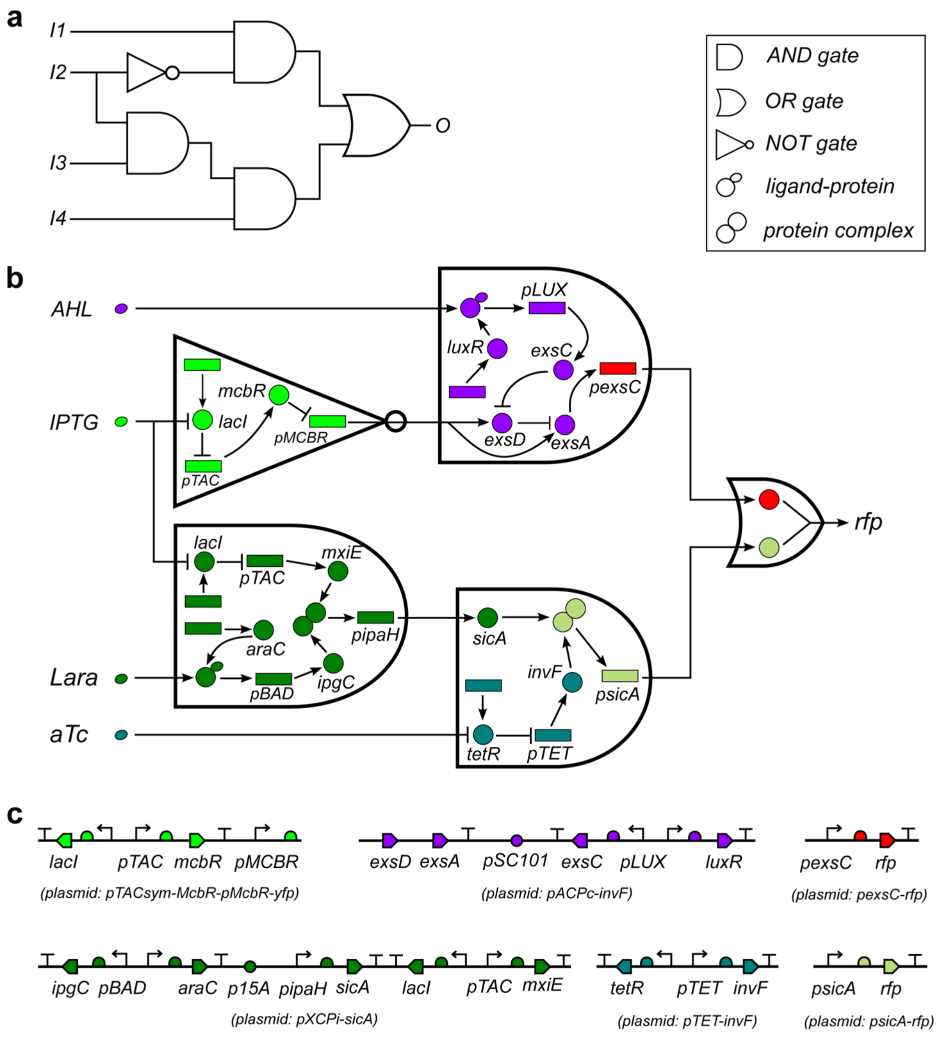


Figure 2. Proposed solution for the circuit D2.[10] Notation shapes for molecular species and genetic parts are used as the same in Figure 1. (a) The input circuit topology. (b) A proposed circuit is represented at the biological level, the mRNA from constitutive promoters is not labeled. (c) A proposed set of 6 modules (with the name of plasmids that these modules are amplified, all plasmids are from refs 12 and 21) to construct the circuit D2 (constitutive promoters are not labeled); they can be assembled into two plasmids (upper row and lower row respectively). Colors are used to show if components belong to the same module or not in parts b and c.

objective is to minimize both the number of cell types and the size of the resulting circuit. In addition, the objective function can be further extended to take into account several assembly methods, including one-pot methodologies. Furthermore, we can also incorporate additional criteria for the module choice. For example, a circuit can contain several AND gates but the role of each gate can be different, since the circuit functionality can be more sensitive to their operation, which can be assessed through critical path and sensitivity analysis. Such addition can lead to circuits with increased tolerance to parameter variation and extrinsic noise. If the cost of a solution can be estimated through the cost of subsolutions then we can apply the methodology of this paper (i.e., merge topologies that share common motifs and utilize the dynamic programming approach with an branch-and-bound extension) in order to reach a solution. Otherwise, we can modify the current algorithm to find a list of candidates first and then we can simulate each of them to select the optimal one based on these new criteria. Finally, the methodology presented here can be extended for higher abstraction levels that would allow the high-level description of complex modules.

The work presented here brings us a step closer to the automated design of robust synthetic circuits. A central challenge that remains is the automated topology selection, since in this work the abstract topology is provided by the user. Toward that goal, we can borrow the multilevel logic optimization from electrical engineering in the case of digital circuits and use heuristic methods for analog dynamics. In addition, in the work presented here, we do not consider environmental and chassis parameters that may be critical to the functionality of the parts, modules, and, hence, final circuit. The existence of characterization data for parts and modules in distinct environments and strains can certainly be taken into account to increase the chances of a successful design.[22] Furthermore, the integration of whole-cell, multi-scale models[23,24] with CAD tools will bring us a step closer to accurately describe the trifecta of circuit, cell and environment. This integrative, computationally-driven approach will lead to a better understanding of how organisms and circuits adapt and respond in dynamic environments.[25,26,27]

METHODS

Definitions. *Abstraction Levels of a Circuit Specification.* We define the following circuit specifications with decreasing abstraction levels (see Figure 1)

- *Functional level* represents the relationship between the input signal information (e.g., concentration, light density, temperature) and the output signal information, but it does not provide information regarding the underlying biological processes that lead to this relationship. For this level, a more detailed representation is possible by introducing intermediate signals and specifying their relationship. This relationship can capture both digital and analog behavior. For example, a circuit can be specified by a truth table for digital behavior or by a function for analog behavior (e.g., $f(x,y) = \ln(x) + \ln(y)$ as in ref 13) or it can be described more detailed by a network of logical gates or a network of computational functions (e.g., positive-logarithm, adder as in ref 13).
- *Biological level* represents how the circuit can process the input signals to produce the output signals through biological processes such as transcription, translation, or other molecular interactions.
- *Physical level* represents the DNA sequence of the circuit. This sequence is also annotated as a linear sequence of genetic parts (e.g., promoter, gene, RBS, terminator) and their relative position (i.e., downstream or upstream of another part).

Since three levels above are arranged by a decreasing abstraction order, one instance in an upper level may have more than one possible corresponding instances in the lower level. One relationship in the functional level can be generated by multiple and distinct biological processes because of the biological diversity. For instance, a logical AND operator of two inputs can be generated through a hybrid promoter,[14,15] a heterodimer formation,[12,16] or through the binding of two RNAs.[17,18] Similarly, a circuit specification at the biological level can have more than one corresponding instances at the physical level by swapping the order of genetic parts or altering the final physical implementation. For instance, both specifications at the physical level pCONST-lacI-pLAC-gfp and pLAC-gfp-pCONST-lacI have the same specification at the biological level. Once a circuit specification is provided at the physical level, the synthetic circuit can be constructed in the lab.

Circuit Graph. To give the user more flexibility to select the abstraction level of the circuit specification that is optimal for the application at hand, the circuit is specified at a mixture between the functional level and the biological level. This specification is represented by the same *circuit graph* definition that was introduced in the SBROME framework.[7] Briefly, each circuit is represented by a circuit graph $G = (V, E, V_I, V_O, \tau_1, \tau_2, \nu)$ where V and E are the vertex set and the edge set, V_I and V_O are subsets of V that represent inputs and outputs respectively, τ_1 is the function that assigns each node with a type which can be either a molecular species (e.g., ligand, mRNA, protein) or a functional device (e.g., AND gate, NOT gate, oscillator) (see Table 1 in ref 7 for more details), τ_2 is the function type for each edge (activatory or inhibitory), and finally the function ν assigns each node with a name (e.g., tetR, lacI). The separation of type and name of a node allows us to increase the abstraction level further by leaving the name of some nodes as "unknown" although their types have to be specified. A circuit graph that contains "unknown" nodes is called an *abstract circuit graph*.

Part and Module Library. Each part and module is a DNA sequence that is specified at the physical level. To build the module library, we collected the description of 102 *E. coli* plasmids. These plasmids were constructed and published in 19 papers. Next, for each plasmid, we enumerated all fragments that contain two or more consecutive parts and we removed any repeated or redundant fragments. Consequently, each fragment (which was represented at the physical level) was converted into a network of molecular species at the biological level. This conversion was done through a simple transformation as in ref 7. Finally, only fragments where the corresponding network has one output were imported as modules into the library. The final library is specified as in Table 2.

Cost Function. The cost of a given solution is defined as the number of PCR and ligation steps required to build the circuit by using the parts/modules in the solution by standard cloning. For that purpose, the cost function can be defined simply by the number of parts and modules since we need one step (i.e., one PCR and one ligation) for each part/module. However, we have to put a reward for modules that are full plasmids since they may be transformed directly into the host cell without amplifying them and ligating them with other parts or modules. More specifically, for a set S of parts and modules, the cost $\mathrm{COST}(S)$ of S is defined as $\mathrm{COST}(S) = \sum_{s\in S} \mathrm{COST}(s)$, where the cost $\mathrm{COST}(s)$ of part/module s is set to $1 - \varepsilon$, with ε being equal to zero in all cases, except when the module exists in a single plasmid. The parameter ε is equal to 0.05 here. By doing that, we favor solutions that do not require the simultaneous presence of multiple plasmids, which can be incompatible or create transformation issues, in the host cell.

Part and Module Selection Problem. Given an (abstract) circuit graph and a library of parts and modules, find a minimum-cost set of parts/modules that can be assembled together to match with the given input circuit graph so that all connections are compatible and in the absence of cross-talk. In this research, we limit our study for only circuits that the topology is a single output directed acyclic graph.

Expansion from Functional Behaviors to Biological Processes. For the circuit and library specifications to be compatible, the user circuit specification is first represented in the corresponding biological level. To do so, each functional device node in the input circuit graph is substituted by a *topology motif* (i.e., a network of interacting components). However, there are more than one substitution for each such node because of the biological diversity, and thus, the number of all possible substitution combinations increases exponentially with the number of functional device nodes. Therefore, a naive approach that considers all these combinations sequentially is very inefficient. Instead, we propose to merge all these combinations into one graph only (Figure 3 illustrates how this is performed by using a NOT-AND circuit as an example), and then, a selection algorithm is applied on the combined graph to find the optimal physical specification. Since many combinations share common topology motifs, this approach improves the performance by avoiding the repeated searching on these common motifs.

Formally, algorithm 1 presents the way to merge all possible topologies for an input circuit graph. For each functional device node, the algorithm searches the motif library to find motifs that are compatible to the functional representation of the device node. A motif is deemed compatible when it corresponds to the same functional behavior and there is a

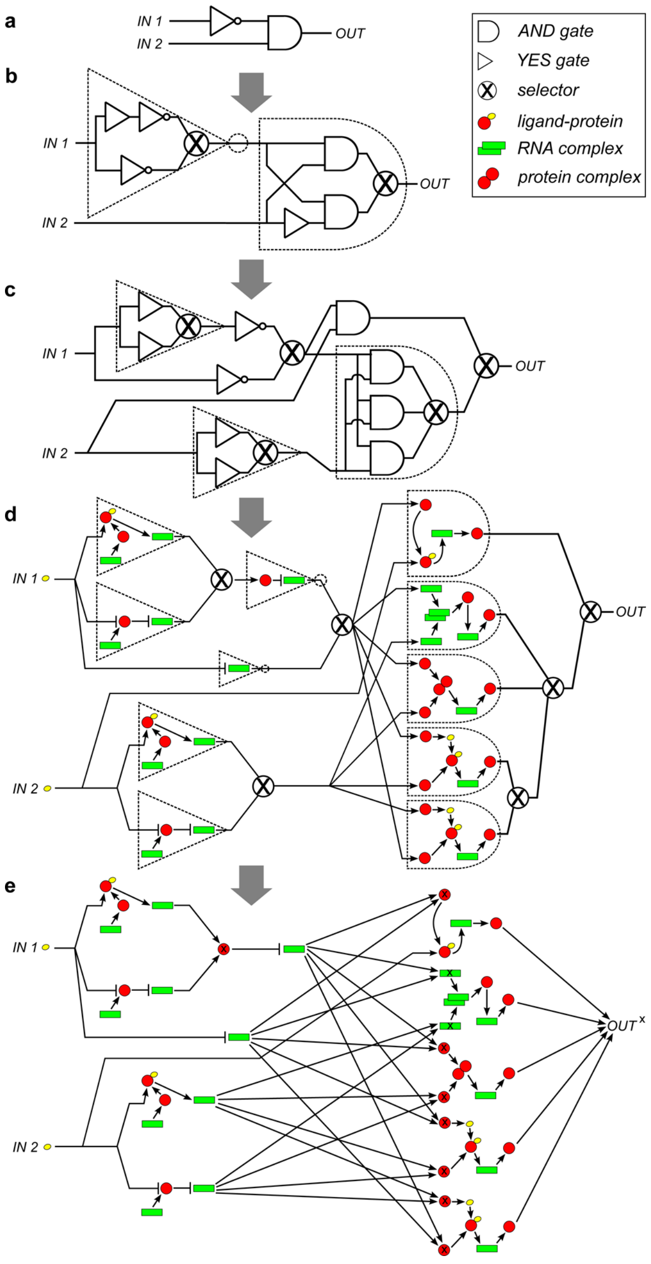


Figure 3. Example on the biological expansion of a NOT-AND circuit. Notation shapes for logical gates and molecular species are used as the same in Figure 1. (a) Functional specification of a NOT-AND gate. (b) Each gate is expanded with two different motifs, one is direct from the ligand input and another is through a YES gate to convert from ligand to mRNA. (c) Each YES gate is expanded with two different motifs while the lower AND gate is expanded with three different motifs, other gates (two NOT gates and the upper AND gate) are not expanded since they have only one motif for their corresponding inputs and outputs. (d) Each gate is expanded to the corresponding biological representations. Note that the last motif of AND gate (through the complex of a ligand and a protein) is not symmetric so it is duplicated to ensure all possible connections are considered. (e) When all seletors are removed, the resulting graph can represent all 27 different topologies for the NOT-AND circuit. Nodes denoted with a cross (×) are new selector nodes.

mapping between its I/O and the I/O of the functional device node. In the current version, the input type and the output type of any circuit are ligand and protein, respectively. When two functional device nodes are connected together, the output type of the first device node (which is also the input type of the second device node) is represented as a mRNA, which is the standard signal carrier for transcription-based devices.[19] Whenever a functional device node is substituted by a set of topology motifs, a selector node is added to merge the output of all these motifs, as proposed in ref 20. However, the introduction of selector nodes prevents the matching between the merged graph with a set of parts and modules in the library since these selector nodes are not part of any module. To avoid that, we simply remove these selectors by expanding all connections that are possible in a given selection. The destination node of these expanded connections is marked as a "selector" node (Figure 3e) and are processed during the part and module selection phase.

DP-Based Algorithm for Part and Module Selection. After the input circuit graph is expanded to encode all possible topologies at the biological level, the resulting graph is matched with parts and modules from the library to identify the optimal set. The selection problem is similar to the topology mapping problem,[9] which can be solved efficiently by using a dynamic programming approach. However, biological circuits are different from electrical ones in at least two important aspects. First, biological circuits use chemical molecules as signals and thus the connection between their components is not always compatible. Second, these circuits have no wiring, thus a module cannot be used more than once in the same circuit because of cross-talk effects. Thus, the optimal subsolutions may be incompatible or with possible cross-talk effects and we cannot combine them together to identify a globally optimal solution. Therefore, the optimal solution sometimes has to be constructed from nonoptimal subsolutions, and this breaks the optimal substructure condition that is essential for the dynamic programming approach.

To overcome these obstacles, we have to store both nonoptimal subsolutions and the information about the connection compatibility and cross-talk effects to ensure that an optimal solution is found. For each node v, a *transitive input subgraph* of v is a subgraph that contains node v and all nodes connected to v by directed paths. A *transitive fan-in subgraph* of v is a connected subgraph of a transitive input subgraph of v that also contains v itself. A *subsolution* of v is a matching of a set of parts and modules with the transitive input subgraph of v that satisfies the constraints of connection compatibility and cross-talk absence. The set $R(v)$ is used to store all possible subsolutions of v and their necessarily related information. In particular, each element $r \in R(v)$ is a 4-component vector that includes (i) the subsolution construction cost, (ii) the output signal of this subsolution (i.e., a molecular species that is matched with v), (iii) all molecular species that are used as signals for this subsolution, and (iv) information about the subsolution construction so that the procedure TRACEBACK can find the way to assemble parts and modules to build this subsolution.

The key idea here is that the set $R(v)$ is constructed recursively following algorithm 2. All nodes are traversed in their topological order (i.e., if there is an edge from node u to node v then node u is traversed before node v). At each node v, the procedure MATCH finds a matching set M in which each element is a matching between a molecular species network (which represents a part or a module at the biological level) with a transitive fan-in subgraph at v. The procedure MATCH also checks to ensure that when a node in the transitive fan-in subgraph is matched, all its incoming edges are also matched,

Algorithm 1 Expanding and merging gene circuit algorithm

```
1: Input: A circuit graph $G = (V, E, \_, \_, \tau_1, \_, \nu)$
2:        A topology motif library $T$
3:        // _ means this variable is skipped since it does not appear in the algorithm
4:        // [ ] denotes extensions
5: Result: All functional device nodes in $G$ are substituted by topology motifs and $G$
   contains molecular species nodes only
6: // Expand & merge
7: while there exists a functional device node $v \in V$ such that $\nu(v)$ is not determined do
8:     $I_v \leftarrow$ all nodes connect to $v$
9:     $O_v \leftarrow$ all nodes connect from $v$
10:    Insert a selector node $s$ into $V$
11:    Insert edges from $s$ to each node in $O_v$ into $E$
12:    $M \leftarrow$ All motifs in $T$ for a node with type $\tau_1(v)$, input $\tau_1(I_v)$ and output $\tau_1(O_v)$
13:    for $m \in M$ do
14:        for each permutation $\sigma$ of INPUT($m$) do
15:            $G \leftarrow$ UNION($G, m$)
16:            Insert an edge (OUTPUT($m$), $s$) into $E$
17:            Insert edges from each node in $I_v$ to a corresponding one in $\sigma$ into $E$
18:            [ if all inputs of $m$ are symmetric then break ]
19:        end for
20:    end for
21:    Remove $v$ from $G$
22: end while
23: // Remove all selector nodes
24: while there exists a selector nodes $s \in V$ do
25:    $I_s \leftarrow$ all nodes connect to $s$
26:    $O_s \leftarrow$ all nodes connect from $s$
27:    for each node $v \in O_s$ do
28:        Mark $v$ as a selector
29:        for each node $u \in I_s$ do
30:            Insert an edge $(u, v)$ into $E$
31:        end for
32:    end for
33:    Remove $s$ from $G$
34: end while
```

except in the case of a selector node where only one incoming edge is necessary to be matched. Next, for each matching $m \in M$, INPUT(m) is a set of circuit graph nodes, each matched with an input node of the network. Finally, any pair of m and a combination of subsolutions (in which each of them is a subsolution of a node in INPUT(m) respectively) that satisfies the constraints of connection compatibility (checked by the procedure COMP) and cross-talk absence (checked by the procedure CROSS) forms a subsolution of v that is added into $R(v)$.

Algorithmic Extensions for Increased Performance. Algorithm 2 can be thought of as an exhaustive search that investigates all possible solutions systematically, but its performance is improved by eliminating solutions that violate the constraints of the connection compatibility and cross-talk absence as early as they occur. However, to explore all subsolutions of a node, we have to consider all subsolution combinations from fan-in nodes. Therefore, the computational cost of this operation at each node can be exponential. To reduce the worst-case complexity, we need to reduce the number of subsolutions that are kept at each node by eliminating unnecessary ones. The following three extensions aim to achieve exactly that.

Detect Symmetric Motif Inputs to Eliminate Duplicated Subsolutions. When a functional device node that has more than one input is substituted by a topology motif, we have to consider all possible connections between preceding nodes (nodes that are connected to this functional device node in the input circuit graph) with the input nodes of the motif (as for the AND gate that uses a ligand-protein complex in Figure 3d). However, in many cases, the input nodes of the motif may be symmetric (i.e., their role is the same), and we only need to consider one representative connection and skip all the other ones (as for the AND gate that uses RNA complex or protein complex in Figure 3d). This improvement (line 18 in algorithm 1) helps the algorithm remove all duplicated subsolutions at the selector node generated by the substitution.

Algorithm 2 Module selection algorithm

```
1: **Input:** A circuit graph $G = (V, \_, \_, \{v_0\}, \_, \_, \_)$
2:         Module & part library $L$
3: **Output:** An optimal circuit construction $OptSol$
4: // Calculate $R(v)$ for all $v \in V$, [ ] denotes extensions
5: **for** each node $v \in V$ in the topological order **do**
6:     [ Find a heuristic solution by apply this Algorithm itself with a threshold $k$
7:     and then update the best found solution cost $c_h$]
8:     $R(v) \leftarrow \emptyset$
9:     $M \leftarrow \text{MATCH}(G, v, L)$
10:    **for** each match $m \in M$ **do**
11:        $\{u_1, u_2, \ldots, u_k\} = \text{INPUT}(m)$
12:        **for** $r = (r_1, r_2, \ldots, r_k) \in \prod_{i=1}^{k} R(u_i)$ **do**
13:            $c \leftarrow \sum_{i=1}^{k} \text{COST}(r_i) + \text{COST}(m)$
14:            $s \leftarrow \bigcup_{i=1}^{k} \text{SIGNAL}(r_i) \cup \text{SIGNAL}(m)$
15:            $o \leftarrow \text{OUTPUT(m)}$
16:            **if** $(\text{COMP}(m, r) \wedge \neg\text{CROSS}(m, r)$ [ $\wedge$ $\text{LOOKAHEAD(o,v)}$ $\wedge$ $c \leq c_h$ ]) **then**
17:                $r' \leftarrow (c, \text{o}, s, (m, r))$
18:                $R(v) \leftarrow R(v) \cup \{r'\}$
19:                [**if** $\exists r'' \in R(v)$ such that $\text{SAMEBIONET}(r', r'')$
20:                    **if** $\text{COST}(r') < \text{COST}(r'')$
21:                        $R(v) = R(v) \setminus \{r''\}$
22:                    **else**
23:                        $R(v) = R(v) \setminus \{r'\}$
24:                    **end if**
25:                **end if**]
26:            **end if**
27:        **end for**
28:    **end for**
29: **end for**
30: // Trace back to find an optimal solution
31: $O = \{r \in R(v_o) \mid \text{COST}(r) \text{ is minimum}\}$
32: $OptSol = \text{TRACEBACK}(O)$
```

Utilize a Connection Look-Ahead Scheme to Prune Subsolution Branches. Algorithm 2 looks ahead at the type of connection in a node to decide if a subsolution at this node can be developed further to a final solution or not (line 16 in algorithm 2). For example, if a node is connected to another node by an activatory edge but all subsolutions at that specific node correspond to it having an inhibitory output, that solution path is abandoned.

Apply Branch and Bound to Eliminate Subsolutions. Although a direct DP application without keeping all nonoptimal subsolutions is not possible, a branch-and-bound scheme can be applied to eliminate all subsolutions that cannot lead to a finally optimal solution. This can be achieved in the following two cases:

- If there are two subsolutions that share the same biological network but have different cost, then if the subsolution with a smaller cost leads to a violation on the constraints of the cross-talk absence or the connection compatibility, so does the subsolution with a larger cost. Therefore, the subsolution with a larger cost can be pruned without losing the optimality (lines 19–25 in algorithm 2).
- Suppose that a nonoptimal solution X has been already found. Let S be a subsolution such that $\text{COST}(S) > \text{COST}(X)$, then any solution X' that is developed from S also contains all parts and modules of S, thus $\text{COST}(X') = \sum_{s \in S} \text{COST}(s) + \sum_{s \in X' \setminus S} \text{COST}(s) \geq \sum_{s \in S} \text{COST}(s) = \text{COST}(S) > \text{COST}(X)$. Therefore, all final solutions that are developed from S are not better than the solution X and thus the subsolution S can be pruned without losing the optimality (lines 6, 7, and 16 in algorithm 2). To prune many such subsolutions S, we

need to find the solution X as soon as possible. Since X does not need to be optimal, we can use a heuristic to find it. To do that, we can modify algorithm 2 itself to create a heuristic algorithm by using a threshold value k to limit the maximum number of subsolutions for each node (i.e., for each node v, $R(v)$ contains at most k best subsolutions ranking by the cost). In Table 3, for the heuristic in the method MMHS and the one for the branch-and-bound extension in the method MMDP, we chose $k = 20$ as it leads to superior performance on this dataset.

■ ASSOCIATED CONTENT

Ⓢ Supporting Information

The source code, part, and module library in SBOL format, benchmarks, an online version of the SBROME tool and a tutorial. 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 acknowledge the support from National Science Foundation (NSF) grant 0941360 and 1146926 and the NSF CAREER grant 1254205 to I.T.

■ REFERENCES

(1) 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.
(2) Pedersen, M., and Phillips, A. (2009) Towards programming languages for genetic engineering of living cells. *J. R. Soc., Interface 6*, S437−S450.
(3) 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.
(4) Marchisio, M. A., and Stelling, J. (2011) Automatic design of digital synthetic gene circuits. *PLoS Comput. Biol. 7*, e1001083.
(5) 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.
(6) 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.
(7) 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.
(8) Purnick, P. E., and Weiss, R. (2009) The second wave of synthetic biology: From modules to systems. *Nat. Rev. Mol. Cell Biol. 10*, 410−422.
(9) Keutzer, K. (1988) DAGON: Technology binding and local optimization by DAG matching. *Proc. DAC '87 Proceedings of the 24th ACM/IEEE Design Automation Conference*, 341.
(10) Rinaudo, K., Bleris, L., Maddamsetti, R., Subramanian, S., Weiss, R., and Benenson, Y. (2007) A universal RNAi-based logic evaluator that operates in mammalian cells. *Nat. Biotechnol. 25*, 795−801.
(11) 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.
(12) Moon, T. S., Lou, C., Tamsir, A., Stanton, B. C., and Voigt, C. A. (2012) Genetic programs constructed from layered logic gates in single cells. *Nature 491*, 249−253.
(13) Daniel, R., Rubens, J. R., Sarpeshkar, R., and Lu, T. K. (2013) Synthetic analog computation in living cells. *Nature 497*, 619−623.
(14) 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.
(15) Ramalingam, K. I., Tomshine, J. R., Maynard, J. A., and Kaznessis, Y. N. (2009) Forward engineering of synthetic biological AND gates. *Biochem. Eng. J. 47*, 38−47.
(16) 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+.
(17) Anderson, J. C., Voigt, C. A., and Arkin, A. P. (2007) Environmental signal integration by a modular AND gate. *Mol. Syst. Biol. 3*, 133.
(18) Callura, J., Cantor, C., and Collins, J. (2012) Genetic switchboard for synthetic biology applications. *Proc. Natl. Acad. Sci. U.S.A. 109*, 5850−5855.
(19) Canton, B., Labno, A., and Endy, D. (2008) Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol. 26*, 787−793.
(20) 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.
(21) Lou, C., Stanton, B., Chen, Y.-J., Munsky, B., and Voigt, C. A. (2012) Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat. Biotechnol. 30*, 1137−1142.
(22) 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.
(23) Tagkopoulos, I., Liu, Y. C., and Tavazoie, S. (2008) Predictive behavior within microbial genetic networks. *Science 320*, 1313−1317.
(24) Karr, J., et al. (2012) A Whole-Cell Computational Model Predicts Phenotype from Genotype. *Cell 150*, 389−401.
(25) Mozhayskiy, V., and Tagkopoulos, I. (2012) Horizontal gene transfer dynamics and distribution of fitness effects during microbial *In silico* Evolution. *BMC Bioinformatics*, DOI: 10.1186/1471-2105-13-S10-S13.
(26) Dragosits, M., Nicklas, D., and Tagkopoulos, I. (2012) A synthetic biology approach to self-regulatory recombinant protein production in *Escherichia coli*. *J. Biol. Eng. 6*, 2.
(27) Dragosits, M., Mozhayskiy, V., Quinones-Soto, S., Park, J., and Tagkopoulos, I. (2013) Evolutionary potential, cross-stress behavior, and the genetic basis of acquired stress resistance in *Escherichia coli*. *Mol. Syst. Biol. 9*, 643.