Recombinase-Based Genetic Circuit Optimization

Chun-Ning Lai, Jie-Hong Jiang, François Fages

▶ **To cite this version:**

Chun-Ning Lai, Jie-Hong Jiang, François Fages. Recombinase-Based Genetic Circuit Optimization. BioCAS 2017 - 13th IEEE Biomedical Circuits and Systems Conference, Oct 2017, Turin, Italy. , pp.1-4, 2017. hal-01659183

HAL Id: hal-01659183

https://inria.hal.science/hal-01659183

Submitted on 8 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Recombinase-Based Genetic Circuit Optimization

Chun-Ning Lai
Grad. Inst. of Electronics Engineering
National Taiwan University, Taiwan

Jie-Hong R. Jiang
Grad. Inst. of Electronics Engineering
National Taiwan University, Taiwan
Email: jhjiang@ntu.edu.tw

François Fages
Inria, Université Paris-Saclay
France

***Abstract*—The rapid advancements of synthetic biology show promising potential in biomedical and other applications. Recently, recombinases were proposed as a tool to engineer genetic logic circuits with long-term memory in living and even mammalian cells. The technology is under active development, and the complexity of engineered genetic circuits grows continuously. However, how to minimize a genetic circuit composed of recombinase-based logic gates remain largely open. In this paper, we formulate the problem as a cubic-time assignment problem and solved by a 0/1-ILP solver to minimize DNA sequence length of genetic circuits. Experimental results show effective reduction of our optimization method, which may be crucial to enable practical realization of complex genetic circuits.**

I. Introduction

Synthetic biology aims at engineering living organisms to behave in some desired manner [1]. It rapid advancements have shown promising applications in biomedicine, bioenergy, and other areas of societal interest. Recombinases, a kind of genetic recombination enzymes common in nature controlling gene expression and modifying genome structures in living organisms, have been exploited as a biotechnology for genetic engineering [2], [3]. A recombinase binds to a DNA strand at its specific target sites with respect to its unique recognition nucleic acid sequences. It may enable excision/insertion, inversion, translocation and cassette exchange on DNA sequences. The inversion function of recombinases has been exploited to construct two-input logic gates in *Escherichia coli* cells with long-term memory [3]. It provides a fundamental tool for logic circuit synthesis using recombinases. In [4], two mechanisms, tyrosine recombinase DNA excision and serine recombinase DNA inversion are exploited to implement simple arithmetic logic units and Boolean functions such as full-adder, substractor, and decoder in mammalian cells. Moreover, with the help of genome editing tools such as CRISPR/Cas9 systems [5], complex genetic circuits have great potential to be implemented in the future. The automation for large scale genetic circuit synthesis and optimization becomes indispensable.

In the prior work [6], a library consisting of 44 recombinase-based gates with up to three inputs is built, and existing logic synthesis tools are adopted to perform optimization and technology mapping for genetic circuit construction. There are several shortcomings of prior approach [6]. Conventional technology mapping algorithms often assume that a logic gate has a single output. In fact, as we shall show later it is natural to consider a genetic logic gate with multiple outputs. Technology mapping under the single-output assumption may yield sub-optimal results.

This work was supported in part by the Ministry of Science and Technology of Taiwan under grant MOST 106-2923-E-002-002-MY3.

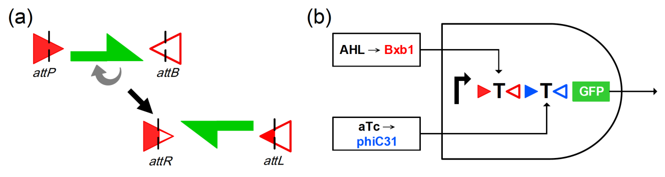

Fig. 1. (a) Recombinase-mediated DNA inversion; (b) a two-input AND gate built from recombinases. [6].

In this work, we aim to overcome the above shortcomings. Our main results include formulating the gate merging problem targeting multi-output gate utilization, building a library of merge-friendly standard cells for logic synthesis of recombinase-based genetic circuits, showing the tractability (cubic time complexity) of gate merging, and solving it with 0/1-integer linear programming (0/1-ILP). Experimental results show promising reduction on the DNA sequence length and level of the synthesized genetic circuits. Note that minimizing the total DNA sequence length is important because a shorter DNA sequence is more likely to succeed in vector insertion into the host cell for the intended computation. On the other hand, minimizing the number of genes and the depth of protein-production cascade is important because protein production in the host cell causes metabolic burden and takes long time due to the additional translation and induction steps. Our methods can be crucial to practical realization of complex genetic circuits for successful applications.

II. Preliminaries

A (combinational) logic circuit is a directed acyclic graph $G(V, E)$ with the set V of nodes and a set $E \subseteq V \times V$ of directed edges. A node $v \in V$ can be a *primary input* (*PI*), *primary output* (*PO*), or a logic gate. For each directed edge $(u, v) \in E$, u is a *fanin* of v, and v is a *fanout* of u. The fanin (resp. fanout) set of a node v is denoted as $FI(v)$ (resp. $FO(v)$). A node with no fanin (fanout) is a PI (resp. PO). A logic gate v is associated with a Boolean function that determines the output value of v depending on the values of its fanins.

A *recognition site pair* associated with a site-specific recombinase is a pair of special DNA sequences, including the *attB* (*attachment site bacteria*) and the *attP* (*attachment site phage*), targeted by the recombinase. When the recognition site pair is bound by the associated recombinase, an irreversible inversion occurs. The DNA subsequence sandwiched by the recognition site pair, including part of the recognition site sequences *attB* and *attP*, is inverted. As a result, the recognition site sequences *attB* and *attP* after the inversion are irreversibly

TABLE I. Some Simple Gates and their DNA Sequences

Gate	Function	DNA Sequence	Cost
CONST0	0	G	1
CONST1	1	PG	2
BUF(a)	a	$\rotatebox[origin=c]{180}{P}G$	2
NOT(a)	$\neg a$	P_aG	2
AND2(a,b)	$a \cdot b$	$\rotatebox[origin=c]{180}{P}_aT_bG$	3
OR2(a,b)	$a \vee b$	$\rotatebox[origin=c]{180}{P}_a\rotatebox[origin=c]{180}{P}_bG$	3
NAND2(a,b)	$\neg a \vee \neg b$	P_aP_bG	3
NOR2(a,b)	$\neg a \cdot \neg b$	$P_a\perp_bG$	3
XOR2(a,b)	$\neg a \cdot b \vee a \cdot \neg b$	$\rotatebox[origin=c]{180}{P}_{ab}G$	2
XNOR2(a,b)	$a \cdot b \vee \neg a \cdot \neg b$	$P_{ab}G$	2
IMPLY(a,b)	$\neg a \vee b$	$\rotatebox[origin=c]{180}{P}_bP_aG$	2
NOTIMPLY(a,b)	$a \cdot \neg b$	$\rotatebox[origin=c]{180}{P}_a\perp_bG$	2
AND$k(v_1,\ldots,v_k)$	$v_1 \wedge \cdots \wedge v_k$	$\rotatebox[origin=c]{180}{P}_{v_1}T_{v_2}\ldots T_{v_k}G$	$k+1$
OR$k(v_1,\ldots,v_k)$	$v_1 \vee \cdots \vee v_k$	$\rotatebox[origin=c]{180}{P}_{v_1}\rotatebox[origin=c]{180}{P}_{v_2}\ldots\rotatebox[origin=c]{180}{P}_{v_k}G$	$k+1$

altered into different products *attR* and *attL*, respectively. As shown in Fig. 1, taken from [6], the inversion mechanism is illustrated in Fig. 1(a), where the triangle pair denotes the recognition site pair.

To see how recombinases can be exploited for logic circuit construction, consider the two-input AND gate shown in Fig. 1(b), where the right-turn arrow denotes a promoter, the two T letters denote terminators, and the green box represents the gene encoding the green fluorescent protein (GFP). (In this paper we read a DNA sequence from left to right assuming the 5'-to-3' direction of the coding strand.) Notice that the first (resp. second) terminator is sandwiched by the recognition site pair of recombinase Bxb1 (resp. phiC31), and recombinases Bxb1 and phiC31 are induced by molecules AHL and aTc, respectively. By the transcription process, the *RNA polymerase* (*RNAP*) binds to the promoter and traverses the DNA template strand until a terminator is encountered. Accordingly, the output GFP can be highly expressed only when both terminators are inverted (i.e., disabled) by their respective control recombinases Bxb1 and phiC31. Because the output GFP is expressed if and only if both input molecules AHL and aTc are of high concentration to induce the production of recombinases Bxb1 and phiC31, the DNA sequence effectively implements a two-input AND gate.

In this work, we consider DNA sequences built by specific DNA segments, referred to as *DNA units*, including promoters, terminators, genes, inverted promoters, inverted terminators, and inverted genes, denoted as P, T, G, $\rotatebox[origin=c]{180}{P}$, $\perp$, and $\rotatebox[origin=c]{180}{G}$, respectively. Table I shows some simple gates and their associated Boolean functions and (abstracted) DNA sequences. Assuming each DNA unit have similar lengths, the cost of a logic gate is reflected in terms of the number of the DNA units in the sequence. Although our subsequent discussion uses this simple cost metric, our methods are valid to other more accurate models. Note also that there may be multiple different DNA sequences associated with the same Boolean function. For example, both sequences $\rotatebox[origin=c]{180}{P}_aT_bG$ and PT_aT_bG implement AND2 gate. However, the former is preferable to the latter because of its lower cost.

III. Multi-Level Circuit Synthesis

Multi-level implementation of genetic circuits can effectively avoid DNA length blow up. The number of logic levels corresponds to the depth of protein-production cascade in a circuit. Under the multi-level implementation, the output gene of a logic gate may correspond to an inducer of its fanout gate. The multi-level structure allows effective logic sharing at the cost of computation delay due to the cascading of protein production.

A. Gate Merging for Circuit Optimization

To embed a genetic circuit in a DNA molecule for vector insertion into a living cell to conduct the desired computation, the well-formed sequences of the constituent logic gates should be concatenated. Because the transcription process of an RNAP would not stop without a terminator, a terminator should be added at the end of the well-formed sequence of each logic gate to avoid undesired interference between logic gates when their sequences are concatenated. In fact, as we discuss below, there are circumstances under which some of these blocking terminators and output genes can be safely removed.

Consider two logic gates u and v with $u \in FI(v)$. Let the sequences of u and v are concatenated with u's on the left and v's on the right. Suppose that u's output gene X corresponds to the inducer x of v's first DNA unit $\rotatebox[origin=c]{180}{P}_x$. When RNAP can traverse through gene X, the inducer x is highly expressed and in turn triggers the inversion of $\rotatebox[origin=c]{180}{P}_x$ to a promoter unit P for v. The effect is the same as letting the RNAP continue its traversal through the first DNA unit of v. On the other hand, when RNAP cannot traverse through gene X, the inducer x is not expressed and the first DNA unit $\rotatebox[origin=c]{180}{P}_x$ of v remains an inverted promoter $\rotatebox[origin=c]{180}{P}$. The effect is the same as ignoring the first DNA unit of v. Effectively, we can remove the blocking terminator of u and the first DNA unit $\rotatebox[origin=c]{180}{P}_x$ of v. Furthermore, if v is the sole fanout of u, that is, inducer x does not control any other DNA unit except for v's first DNA unit, then gene X can be removed from the sequence of u. For example, let u be a logic gate with output $X = \text{NAND2}(a,b)$ and v be a gate with output $Y = \text{AND2}(x,c)$, where X is the gene of inducer x. Let the DNA sequences of u and v be P_aP_bXT and $\rotatebox[origin=c]{180}{P}_xT_cYT$, respectively. Then the combined DNA sequence can be simplified as follows.

$$\underbrace{P_aP_bX\overbrace{T}^{\text{drop}}}_{X=\text{NAND2}(a,b)}\ \underbrace{\overbrace{\rotatebox[origin=c]{180}{P}_x}^{\text{drop}}T_cYT}_{Y=\text{AND2}(x,c)} \quad \xrightarrow{\text{merge}} \quad P_aP_bXT_cYT$$

$$\xrightarrow{\text{if } |FO(X)|=1} \quad P_aP_bT_cYT$$

This simplification reduces the DNA sequence by two to three DNA units (depending on whether inducer x is used elsewhere) and reduces one logic level in the cascading of gate u to gate v. The above observation leads to the following optimization problem.

Problem Statement 1: Given a logic circuit $G(V,E)$, find a well-formed sequence implementation for each logic gate of V, and merge gates u and v with $(u,v) \in E$ such that the total DNA sequence length and the depth of protein-production cascade are minimized.

B. Gate-Merging Algorithm

We propose an algorithm to exactly solve a simplified version of Problem 1, assuming the DNA sequences of the logic gates are already given and focusing on sequence length minimization. As will be seen in the experiments, the objective of sequence length minimization also effectively reduces the depth of protein-production cascade. Given a combinational circuit $G(V, E)$ with gates $V = \{g_1, g_2, \ldots, g_n\}$ and edges $E \subseteq V \times V$. (Here we exclude PIs and POs from V.) We generate a mergeability graph $G'(V, E')$, where $E' \subseteq E$ for an edge $(g_i, g_j) \in E'$ if and only if gate g_i can be combined with gate g_j. The graph $G'(V, E')$ denotes the mergeability relation over the logic gates. A path $\{N_0, N_1, \ldots, N_k\}$ in the graph $G'(V, E')$ signifies the feasibility of merging N_0 with N_1, then merging with N_2, and so on, and finally merging with N_k. That is, the logic gates $N_0, N_1, \ldots, N_k$ are merged into a single multi-output genetic gate. Hence the optimization problem is equivalent to the *weighted path covering problem* [7], that is, finding a set of disjoint paths that covers all the nodes of $G'(V, E')$ while the total cost of the paths is minimized.

The path covering problem can be transformed to the *minimum assignment problem* [8] and can be solved in cubic time using the Hungarian algorithm [9]. However, we reformulate it as a 0/1-ILP problem to take advangtage of the highly engineered modern ILP solvers. Let x_i be a Boolean variable to indicate whether a gate g_i is *not* merged with one of its fanouts. The gate g_i is not merged with one of its fanouts if and only if variable x_i valuates to 1. For each gate g_i, its cost $C(g_i)$ can be derived as follows by the discussion in Section III-A.

$$C(g_i) = \begin{cases} 2 + |FI(g_i)|, & x_i = 1 \\ |FI(g_i)|, & x_i \neq 1 \wedge |FO(g_i)| \neq 1 \\ -1 + |FI(g_i)|, & x_i \neq 1 \wedge |FO(g_i)| = 1 \end{cases} \quad (1)$$

where $|FI(g_i)|$ (resp. $|FO(g_i)|$) denotes the number of fanins (resp. fanouts) of gate g_i in the circuit. Also we use a Boolean variable $x_{i,j}$ to denote whether g_i is merged with its fanout g_j. The gate g_i is merged with its fanout g_j if and only if variable $x_{i,j}$ valuates to 1. We impose the condition that a gate can only be merged with at most one of its fanouts; also a node can only be merged with at most one of its fanins. These conditions translate to the following 0/1-ILP formulation.

$$\text{minimize} \quad \sum_{g_i \in V} C(g_i) \quad (2)$$

$$\text{subject to} \quad x_i + \sum_{g_j \in FO(g_i)} x_{i,j} = 1, \text{ for } i = 1, \ldots, n \quad (3)$$

$$\sum_{g_i \in FI(g_j)} x_{i,j} \leq 1, \text{ for } j = 1, \ldots, n \quad (4)$$

Under this formulation, there are $2n$ constraints, and $n + \sum_{g_i \in V} |FO(g_i)|$ variables, where n is the number of logic gates in the circuit. Notice that the simplicity of the constraints of Eq. (3) and (4) allows very efficient solving as will be evident in the experiments.

To illustrate, consider the circuit of Fig. 2(a). We are given the DNA sequences of the gates as follows.

$$\underbrace{d_a T_b G_1 T}_{G_1} \underbrace{d_c d_d G_2 T}_{G_2} \underbrace{P_{g_1} G_3 T}_{G_3} \underbrace{d_{g_1} T_{g_2} G_4 T}_{G_4} \underbrace{d_e G_5 T}_{G_5}$$
$$\underbrace{d_{g_3} L_{g_4} G_6 T}_{G_6} \underbrace{d_{g_4} G_7 T}_{G_7} \underbrace{d_{g_4} T_{g_5} G_8 T}_{G_8}$$

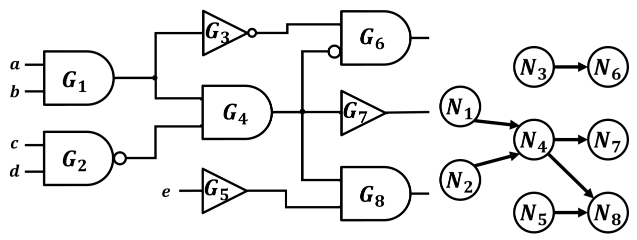

Fig. 2. (a) Circuit under gate merging. (b) Mergeability graph.

where G_i and g_i denote the output gene and its corresponding inducer, respectively, of gate G_i. The corresponding mergeability graph of the circuit is shown in Fig. 2(b). Before gate merging, the total cost is 29 DNA units. To perform gate merging, the following 0/1-ILP instance is formed.

$$\begin{aligned} \text{minimize} \quad & 2x_{1,4} + x_{2,4} + 2x_{4,7} + 2x_{4,8} + \\ & 4x_1 + 4x_2 + 3x_3 + 4x_4 + \\ & 4x_5 + 4x_6 + 3x_7 + 4x_8 \\ \text{subject to} \quad & x_1 + x_{1,4} = 1 \\ & x_2 + x_{2,4} = 1 \\ & x_3 + x_{3,6} = 1 \\ & x_4 + x_{4,7} + x_{4,8} = 1 \\ & x_5 + x_{5,8} = 1 \\ & x_6 = 1 \\ & x_7 = 1 \\ & x_8 = 1 \\ & x_{1,4} + x_{2,4} \leq 1 \\ & x_{4,8} + x_{5,8} \leq 1. \end{aligned}$$

The optimum solution $x_{2,4}, x_{4,7}, x_{5,8}, x_{3,6}, x_1, x_6, x_7, x_8 = 1$ and others 0 is found. It corresponds to four paths $\{N_2, N_4, N_7\}$, $\{N_5, N_8\}$, $\{N_3, N_6\}$ and $\{N_1\}$ with minimized cost 18 DNA units. Hence the cost is reduced by 38%. The corresponding DNA sequence of the genetic circuit is as follows

$$\underbrace{d_a T_b G_1 T}_{G_1} \underbrace{P_c P_d T_{g_1} G_4 G_7 T}_{G_2, G_4, G_7} \underbrace{P_{g_1} L_{g_4} G_6 T}_{G_3, G_6} \underbrace{d_e T_{g_4} G_8 T}_{G_5, G_8}.$$

Notice that in the above merging of gates $G2$, $G4$ and G_7, we exploit the fact that G_4 is input symmetric, and rewrite the sequence $d_{g_1} T_{g_2} G_4 T$ to $d_{g_2} T_{g_1} G_4 T$ for improved merge with G_2 sequence.

C. Overall Synthesis Flow

The overall synthesis flow of recombinase-based genetic circuits is shown in Fig. 3. Given an input circuit, it first undergoes technology-independent optimization and then technology mapping. After technology mapping, the exact gate-merging algorithm presented in Section III-B is applied to minimize the total sequence length.

To maximize the chance of merging gates, we construct a library including CONST0, CONST1, BUF, NOT, AND2, AND3, AND4, AND5, OR2, OR3, OR4, OR5, IMPLY and NOTIMPLY for technology mapping. The library cells mainly have the functions that are associated with sequences started

TABLE II. RESULTS OF CIRCUIT OPTIMIZATION

Circuits	#Gate	#PI/#PO	[6]		Ours					
			Opt. Length	Opt. Level	Org. Length	Opt. Length	Opt. Level	CPU Time	#Var.	#Cst.
b10	151	28/23	460 (1.00)	11 (1.00)	624	410 (0.89)	9 (0.82)	0.00	496	302
b11	418	38/37	1454 (1.00)	25 (1.00)	1746	1155 (0.79)	17 (0.68)	0.00	1365	836
b12	881	126/125	3062 (1.00)	15 (1.00)	3646	2598 (0.85)	10 (0.67)	0.00	2890	1762
b13	220	63/63	725 (1.00)	12 (1.00)	707	640 (0.88)	7 (0.58)	0.00	735	440
b14	3982	277/299	12649 (1.00)	124 (1.00)	16426	11067 (0.88)	41 (0.33)	0.06	12743	7964
b17	18925	1452/1512	68414 (1.00)	104 (1.00)	79782	58268 (0.85)	57 (0.55)	0.31	62369	37850
b18	52078	3357/3343	187906 (1.00)	137 (1.00)	216853	151473 (0.81)	94 (0.69)	0.90	168118	104156
b20	8045	522/512	26735 (1.00)	128 (1.00)	28767	22379 (0.84)	49 (0.38)	0.12	25688	16090
b21	8105	522/512	27070 (1.00)	121 (1.00)	28925	22558 (0.83)	43 (0.36)	0.11	25875	16210
b22	12124	767/757	40711 (1.00)	124 (1.00)	43480	33652 (0.83)	53 (0.43)	0.17	38594	24248

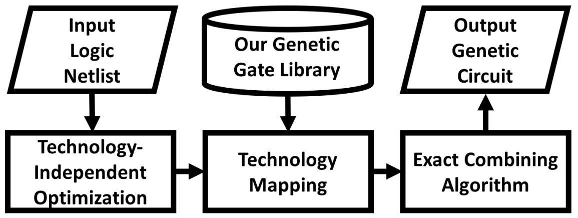

Fig. 3. Optimization flow.

with a reversed promoter d controlled by some inducer. Note that the sequence of NOT gate does not start with d and cannot be combined with other gates. However it is needed for the completeness of technology mapping. Note that IMPLY and NOTIMPLY gates are not input symmetric, i.e., their two inputs are order dependent. For IMPLY(a,b), only the fanin gate at input b can be combined with the IMPLY gate; for NOTIMPLY(a,b), only the fanin gate at input a can be combined with the NOTIMPLY gate. Note also that the costs in Table I differ from those in our library by 1 due to the inclusion of a blocking terminator at the end of each gate sequence as discussed in Section III-A.

IV. Experimental Evaluation

Our algorithm was implemented in the Berkeley synthesis and verification tool, ABC [10], while CPLEX [11] was adopted for 0/1-ILP solving. The experiments were conducted on a Linux machine with a Xeon 3.4 GHz CPU and 32 GB RAM. We compared the circuits synthesized by our method against those synthesized by prior work [6]. Notice that the library of [6], which consists of 44 standard cells with Boolean functions associated with 3-variables bwfs, omits the cost of output genes and blocking terminators (discussed in Section III-A). To have fair comparison, we recalculated the circuit cost of [6]. Essentially for each logic gate, an extra cost of 2 has to be added.

Table II compares our synthesis method with prior work [6]. In the table, Column 1 lists the circuit name; Column 2 shows the gate count; Column 3 shows the numbers of PIs and POs, Columns 4 and 5 show the number of sequence units and the largest level of required protein-production cascade in the synthesized DNA sequence of [6]; Columns 6, 7, 8, 9, 10, and 11 show the number of sequence units after technology mapping and before the combining operation, the number of sequence units after the combining operation, the largest level of required protein-production cascade, the total runtime in CPU seconds, the number of ILP variables, and the number of ILP constraints, respectively, of our method. Note that our library used in technology mapping is more constrained than prior library [6]. Our method tries to combine gates after technology mapping to further reduce the length of DNA sequence and the delay of output protein production. Although the sequence lengths after technology mapping (Column 6) are worse than those of prior work (Column 4), the following combining operation effectively reduces the lengths (Column 7) and achieves 11–21% DNA sequence length reduction compared to [6]. As a by-product, it also achieves 18–67% reduction in the level of protein-production cascade. For the computation time, most of the runtime was spent on ILP solving, and the entire computation takes less than a second for every benchmark circuit.

References

[1] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in : Escherichia coli. *Nature*, 403(6767): 339-342, 2000.

[2] N. Roquet, A. P. Soleimany, A. C. Ferris, S. Aaronson, and T. K. Lu. Synthetic recombinase-based state machines in living cells. *Science*, 353(6297): 363-377, 2016.

[3] P. Siuti, J. Yazbek, and T. K. Lu. Synthetic circuits integrating logic and memory in living cells. *Nature Biotechnology*, 31(5): 448-452, 2013

[4] B. H. Weinberg, N. T. H. Pham, L. D. Caraballo, T. Lozanoski, A. Engel, and S. Bhatia and W. W. Wong. Large-scale design of robust genetic circuits with multiple inputs and outputs for mammalian cells. *Natural Biotechnology*, doi:10.1038/nbt.3805, 2017.

[5] P. S. Choi and M. Meyerson. Targeted genomic rearrangements using CRISPR/Cas technology. *Natural Communications*, 5: 3728, 2014.

[6] T.-Y. Chu and J.-H. R. Jiang. Logic synthesis of recombinase-based genetic circuits. *Scientific Reports*, to appear.

[7] R. Diestel. *Graph Theory*, 3rd ed., Springer, 2005.

[8] D. F. Votaw, and A. Orden. The personnel assignment problem. In *Proc. Symposium on Linear Inequalities and Programming*, pp. 155-163, 1952.

[9] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logistics Quart.*, 2, pp. 83-97, 1955.

[10] R. Brayton and A. Mishchenko. ABC: An Academic Industrial-Strength Verification Tool. In *Proc. Int'l Conf. on Computer Aided Verification*, 2010.

[11] IBM. ILOG CPLEX Optimization Studio. [Online]. Available: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/