Stoichiometrically Minimal Source Pathways via Model Checking

Matthew Fong
EECS, UC Berkeley
mfong92@berkeley.edu

Sanjit A. Seshia
EECS, UC Berkeley
sseshia@eecs.berkeley.edu

ABSTRACT

We formulate the problem of finding stoichiometrically minimal source pathways (SMSPs) in biochemical metabolic graphs and present a model checking approach to solve it. SMSPs are paths whose source nodes correspond to native metabolites and which use a non-dominated amount of those compounds. Our approach allows one to eliminate inefficient pathways when selecting the best path to a target. We also investigate the impact of the choice of model checking technique on the runtime for our procedure.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*; B.7.2 [**Design Aids**]: Verification

General Terms

Algorithms, Design, Verification

1. INTRODUCTION

Enzymatic pathway synthesizers (e.g., [7]) are capable of constructing pathways to target chemicals based on naturally-known reactions as well as reactions that are inferred as plausible. However, few have the capability to provide information on which pathways are better than others, in terms of success likelihood and efficiency. In our work, we make strides towards this goal by defining the problem of finding stoichiometrically minimal source pathways (SMSPs). These SMSPs are defined by their usage of native metabolites, which are compounds biosynthetically accessible from raw sources (e.g., glucose, ammonia, sulfate, and phosphate) using only the enzymes genetically encoded within the host organism. Informally, SMSPs are paths whose source nodes correspond to native metabolites and which use a *non-dominated* amount of those compounds. The SMSP problem is related to that of balancing a metabolic pathway: the latter is a limiting case of the SMSP problem where the coefficients for all cofactors are zero. For many metabolic targets, there are no such balanced paths, but there may be routes that use significantly fewer native metabolites than others.

Problem Statement: We are given a metabolic graph $G = (V, S, U, t)$, where V is the set of all chemicals, $U \subseteq V$ is the set of native metabolites, t is the target, and S is the matrix expressing the coefficients of the reactions, where S_{ji} is the coefficient for chemical i in reaction j. Let $|V| = n$ and $|U| = k$. Cost vector $c = [c_1, \ldots, c_k, \ldots, c_n]$ represents the amount of each chemical that is used (positive) or produced (negative) in any path. For each hyperedge j that is traversed, decrement c_i by S_{ji}. The solution (SMSPs) for a given G is the set of all non-dominated paths from U to t where the cost of a path is represented by the vector $c = [c_1, \ldots, c_k, 0, \ldots, 0]$. As this graph is completely connected, there is some ordering of reactions that force entries of the cost vector to be zero for the non-native metabolites, as we are interested in the stoichiometric cost in terms of only the native metabolites. Path x dominates path y if $c_i^{(x)} \leq c_i^{(y)}\ \forall i \in U$, and $c_j^{(x)} < c_j^{(y)}$ for at least one j, where $c_i^{(x)}$ and $c_i^{(y)}$ are the costs for the ith chemical in paths x and y, respectively. An example of this formalization can be seen in Section 2.

Finding SMSPs can easily be shown to be NP-complete [4] with a reduction from the set partition problem [5], so we take a model checking approach to solve this SMSP problem. In this work, we describe the model checking approach that we employ, as well as results from an *E. coli* system, provided by the Act Ontology pathway synthesizer [7].

2. MODEL CHECKING FORMULATION

For model checking, the state of the system is defined by a vector $q = [\mathrm{N}_1, \ldots, \mathrm{N}_n, rxn]$, where N_i represents the number of units of compound i, and *rxn* represents the reaction in the metabolic graph that was most recently traversed ("fired"). The edges of the metabolic graph define the transition function. The initial state, q_0, is $[0, \ldots, 0]$, where the last 0 denotes that no previous reaction was traversed. The target chemical compound defines the target state that must be reached within a finite number of steps. There are limits on each of the N variables, where M is set to be the maximum number of units of a compound (and should be large enough so that it is never reached). *rxn* can take on the value of any reaction, and can transition to any other valid reaction whose reactants are available. Our approach is illustrated with an example below.

Chemical coefficients ($\mathrm{N}_1, \ldots, \mathrm{N}_n$) increase when a reaction is fired that produces the chemical, and decrease when a reaction uses the chemical. Below, we define S, U, and t for a simple example:

$$2\mathrm{A} + \mathrm{B} \rightarrow \mathrm{C}\ (\mathrm{rxn1}) \qquad \mathrm{B} + 2\mathrm{C} \rightarrow \mathrm{D}\ (\mathrm{rxn2})$$

$$U = \{\mathrm{A}, \mathrm{B}\}, \qquad t = \{\mathrm{D}\}, \qquad S = \begin{bmatrix} -2 & -1 & 1 & 0 \\ 0 & -1 & -2 & 1 \end{bmatrix}$$

The number of units of A, B, C, D are denoted by N_1, N_2, N_3, N_4, respectively. The corresponding transition function for N_3 is given below in the notation of the NuSMV model checker [3]:

```
next(N3):=
case
    N3 > M : N3;          //stop at upper bound
    N3 < -M : N3;         //stop at lower bound
    N4 > 0 : N3;          //stop at target
    rxn = 1 : N3 + 1;     //transition for rxn1
    rxn = 2 : N3 - 2;     //transition for rxn2
    TRUE : N3             //else, is the same
esac;
```

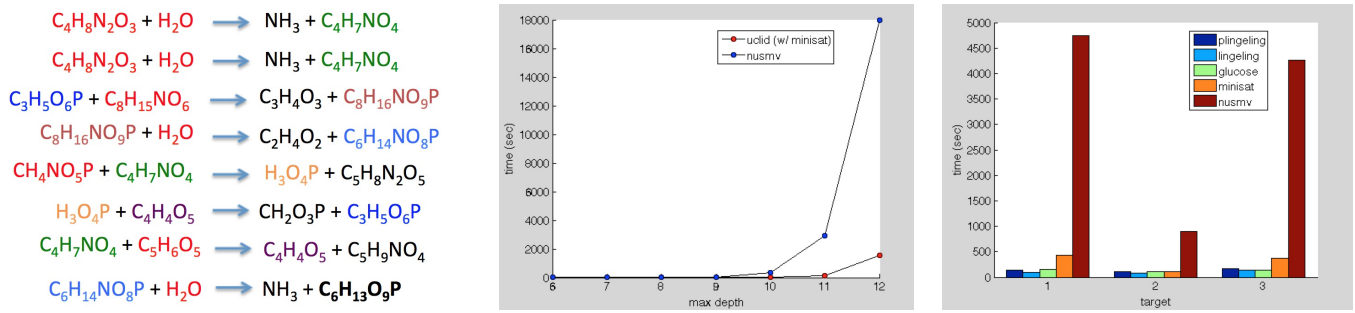

Figure 1: (a) Sample pathway to $C_6H_{13}O_9P$. Red compounds are native. (b) Runtime vs. max depth for finding paths for $C_6H_{13}O_9P$ (c) Comparison of SAT solvers after encoding with UCLID, depth 11 (targets in (c): (1) $C_5H_{11}O_7P$, (2) $C_7H_5NO_4$, (3) $C_3H_7O_6P$).

An appropriate linear temporal logic (LTL) [6] specification must then be checked to generate a counterexample that provides one possible path. We iteratively add a constraint to the LTL formula to eliminate this generated counterexample and generate a new counterexample that is not dominated by this original one (or it returns "no counterexample found"). The characteristics that the counterexample must have are: (1) positive coefficient for target, (2) non-negative coefficients for all non-source chemicals, and (3) not dominated by any previously generated counterexample.

Here is an example on another small graph, with target N51. This first LTL specification has characteristics (1) and (2), and is the initial specification that is evaluated.

```
G (N51 = 0 | (N51 = 0 | (N7 < 0) | (N20 < 0)
| (N28 < 0)))
```

The following counterexample is found: N3 = -1, N5 = -1, N51 = 1, implying that one unit each of Chemicals 3 and 5 can be consumed to produce one unit of 51. The next specification is then:

```
G (N51 = 0 | (N51 = 0 | (N7 < 0) | (N20 < 0)
| (N28 < 0)) | (N3 <= -1 & N5 <= -1))
```

This continues until no further counterexamples are found.

3. RESULTS

First, we examine a concrete solution to this problem. We are using a metabolic graph for *E. coli* generated by Act [7], which has a total of 1253 reactions and 762 chemicals. In Figure 1(a), we see an example of one non-dominated pathway to d-allose-6-phosphate. Although this example only has compounds with coefficient 1, our system accounts for non-unitary coefficients as well. In this case, the cost of this pathway would be the total of the compounds in red on the reactants side. All other reactants in this pathway come from the products of some other reaction in the pathway.

We begin by encoding our model with NuSMV [3], a symbolic model checker. There are two important points to note in this analysis. First, we must impose a maximum search depth for our model checker. In practice, the most interesting (and best) paths occur within a relatively low maximum depth, slightly greater than the depth of the target to account for pathways that are not completely in series. The main tradeoff that we work with is the exponential nature of runtime vs. search depth, as we can see in Figure 1(b). In addition, we report the amount of time that it takes for a given model to reach a result of "no counterexample found", as that signifies the end of the search of the entire state space.

With the slow performance of NuSMV at bounds greater than 10, we attempted to use other model checking techniques to solve this problem. In particular, we used UCLID [1, 2], a model checker based on satisfiability modulo theories (SMT) solving. As shown in Figure 1(b), UCLID dramatically improved the runtime. Moreover, UCLID can be used with any back-end Boolean satisfiability (SAT) solver, and varying the SAT solver paired with UCLID yields further improvements in runtime as shown in Figure 1(c).

In this *E. coli* pathway, for the targets beyond depth 3, there was an average speedup factor of 25x, with a maximum of 60x and a minimum of 8x when comparing NuSMV with the standard UCLID solver. We have run this procedure on 19 different targets with UCLID, and for a bound of 11, the average runtime for these is 161 seconds, with a minimum of 11 seconds. Some examples of other chemical targets we analyzed are d-glucosamine-6-phosphate and 3-hydroxypropionaldehyde, which is a component of the antimicrobial compound Reuterin. We are further investigating the quality of pathways past a certain reaction threshold, as well as other methods to speed up our runtime. More details are available in [4].

4. CONCLUSIONS

We have defined the SMSP problem and shown a viable method of finding SMSPs, with the ability to generate all possible SMSPs (bounded by an input search depth). From our experiments, we already see that the choice of model checker can have a large impact on the runtime, which indicates that further optimization could make greater depths more tractable. Future work can address the variance among the average runtimes for different targets, as well as apply similar methods to other optimal path problems in synthetic biology [4].

Acknowledgment: We are grateful to Chris Anderson and Saurabh Srivastava for their advice and assistance throughout this project.

5. REFERENCES

[1] UCLID Verification System. Available at `http://uclid.eecs.berkeley.edu`.

[2] R. E. Bryant, S. K. Lahiri, and S. A. Seshia. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions. In *CAV*, LNCS 2404, pages 78–92, July 2002.

[3] A. Cimatti, E. M. Clarke, E. Giunchiglia, and et al. Nusmv 2: An opensource tool for symbolic model checking. In *CAV*, pages 359–364, 2002.

[4] M. Fong. Two optimal path problems in synthetic biology. Master's thesis, EECS Department, University of California, Berkeley, May 2015.

[5] N. Karmarkar and R. M. Karp. The differencing method of set partitioning. Technical report, Berkeley, CA, USA, 1983.

[6] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.

[7] S. Srivastava, J. Kotker, S. Hamilton, P. Ruan, J. Tsui, J. C. Anderson, R. Bodik, and S. A. Seshia. Pathway synthesis using the Act ontology. In *Proceedings of the 4th International Workshop on Bio-Design Automation (IWBDA)*, June 2012.