Efficient algorithms for querying enzymes to manipulate the steady state of metabolic pathways

Bin Song, Tamer Kahveci, and Sanjay Ranka

Computer and Information Science and Engineering,
University of Florida, Gainesville, FL, 32611
{bsong,tamer,ranka}@cise.ufl.edu

Abstract. Metabolic pathways show the complex interactions among enzymes that transform chemical compounds. The state of a metabolic pathway can be expressed as a vector, which denotes the yield of the compounds or the flux in that pathway at a given time. The steady state is a state that remains unchanged over time. Altering the state of the metabolism is very important for many applications such as biomedicine, bio-fuels, food industry and cosmetics. The goal of the enzymatic target identification problem is to identify the set of enzymes whose inhibitions lead the metabolism to a state that is close to a given desired or goal state. Given that the size of the solution space is exponential in the number of enzymes, the target identification problem is very computationally intensive.

Given a metabolic pathway and a desired goal state of that pathway, we develop efficient algorithms to solve the enzymatic target identification problem in this paper. We develop a measure, State Distance (SD), that evaluates the affect of the inhibitions of a set of enzymes as a function of the deviation of the steady state of the pathway after their inhibitions from the goal state. Using this measure, we develop two algorithms. The first one is a traversal approach that explores possible solutions in a systematic way using a branch and bound method. The second one uses genetic algorithms to derive good solutions from a set of alternative solutions iteratively. Unlike the former one, this one can run for very large pathways.

Our experiments show that our algorithms can identify enzyme sets whose inhibitions produce steady states that follow the results obtained in vitro in the literature from a number of applications. They also show that the traversal method is a good approximation of the exhaustive search algorithm and it is up to 11 times faster than the exhaustive one. This algorithm runs efficiently for pathways with up to 30 enzymes. For large pathways, the genetic algorithm can find good solutions in less than 10 minutes.

Availability: All the datasets used in this paper and the code developed for this paper are available at `www.cise.ufl.edu/~bsong/ResearchProject/Impact.htm`.

Key words: Metabolic pathways; metabolic engineering; steady state.

1 Introduction

Metabolic pathways are one of the most important data resources in biology. A metabolic pathway is a complex network of chemical reactions occurring within a cell. Enzymes catalyze these reactions. Reactions transform a set of compounds into another set of compounds. Note that, the term "network" is also used in the literature to denote the union of all pathways of an organism or large pathways. To keep the notation consistent, we will use the term pathway instead of network regardless of the pathway size in this paper. The *state* of a metabolic pathway can be expressed as a vector, which denotes the yield of the compounds [27] or the flux [17] in the pathway at a given time. *Steady state* is the state that remains unchanged over time.

Many applications require altering the steady state of a given pathway. For example, a healthy metabolism keeps the state of the pathway at desired levels. External factors or genetic mutations can change the production rate of a set of enzymes. They can even modify the structure of produced enzymes. Such unexpected enzyme behaviors can lead to aberrations in the metabolism. For instance, low or missing activity of an enzyme may result in the blockage of the pathway. Furthermore, this can propagate to other parts of the pathway that need the compounds produced in the blocked part of the pathway. As a result, the production of compounds may increase or decrease. Such aberrations in the state of the metabolism can lead to severe diseases. Examples include mental retardation, seizures, decreased muscle tone, organ failure and blindness [22, 7]. Thus, changing the state of the metabolism back to a desired level is often needed.

Many applications in fields such as cosmetics and food industry require manipulating the state of pathways. For example, fatty acid biosynthesis pathway converts fatty acids that are used in the cosmetic industry in creams and lotions. Butanoate metabolism produces poly-β-hydroxybutyrate which is essential for producing plastics [3]. Mevalonic acid pathway and MEP/DOXP pathway produce carotenoid that are often used as anti-oxidant in food industry [21]. The metabolisms of many organisms, such as bacteria, algae and plants naturally produce these compounds. A common practice is to extract them from these organisms. By manipulating the pathways of these organisms, the production of

these compounds can be increased significantly. Currently, this is done through cutting the consumption of the desired compound by the underlying organism. This strategy, however, is inefficient. There is a great potential to improve the efficiency through accurate computational methods.

One way to change the state of the pathway back to the desired one is to use chemical compounds (i.e. drugs) to inhibit a set of enzymes. When an enzyme is inhibited, it cannot catalyze the reactions it is responsible from. As a result, some entries in the steady state of the pathway may increase and some may decrease. The *Enzymatic Target Identification Problem* aims to identify the set of enzymes whose inhibitions lead to a steady state of the metabolic pathway that is as close to a user supplied goal state as possible. We consider the enzymatic target identification problem in this paper.

The size and the complex structure of the metabolic pathways along with the large size of the solution state space makes the enzymatic target identification problem computationally challenging. We can compute the steady state of a metabolic pathway after inhibiting a given set of enzymes in polynomial time. (See Appendix for a discussion on steady state computation.) However, enzymatic target identification aims to solve the inverse problem of finding the set of enzymes to achieve a steady state that is close to a given goal state. We have shown that a simplified version of the enzyme identification problem is an NP-complete problem [13]. Thus, exhaustive search is impractical for the typical pathways that contain hundreds of enzymes, thousands of compounds and reactions. Assuming that finding a steady state of a pathway takes on the order of 10 milliseconds, it will require nearly a year of computational time to test all solutions with up to four enzyme combinations on a metabolic pathway with 500 enzymes. Existing methodologies restrict the search to a small number of choking points (i.e., enzymes with many connections) or immediate precursors of the targeted compounds. *Targeted compounds* are the ones whose productions need to be stopped. These approaches, however, can have significant deviations from the optimal result. This is because inhibiting a precursor enzyme or a choking point can affect the production of many other compounds that are not targeted.

Sridhar et. al and Song et. al considered a simplified version of the enzymatic target identification problem [24, 25, 23]. In their version, each entry of the state denotes whether the corresponding compound is present or not. For this simplified version, the goal, then, is to identify the set of enzymes whose inhibitions eliminate all the targeted compounds while incurring minimum damage. They have defined *damage* as the number of non-targeted compounds that are eliminated because of inhibitions. Sridhar et al., used a branch and bound strategy to solve this problem for pathways with up to 32 enzymes in less than an hour [25]. Sridhar et. al and Song et. al developed an iterative heuristic method to scale their solutions to large pathways [24, 23]. The binary model described in these works, however, is limited. It cannot address partial production of the compounds. In addition, it ignores the change in flux or yield due to the inhibitions of a set of enzymes.

Figure 1 illustrates the limitations of the binary model used by Sridhar et. al and Song et. al [24, 25]. Inhibiting E_1 knocks out compounds C_2, C_4 and C_5. Suppose C_4 is the targeted compound. Inhibiting E_1 stops two non-targeted compounds (C_2 and C_5). The damage is two using the binary model. There are several drawbacks. One is that the inhibition of E_1 accumulates C_1. The binary model, however, ignores this. Furthermore, although the inhibition of E_1 does not fully eliminate C_7, it influences the production of C_7. The binary model disregards this influence as well.

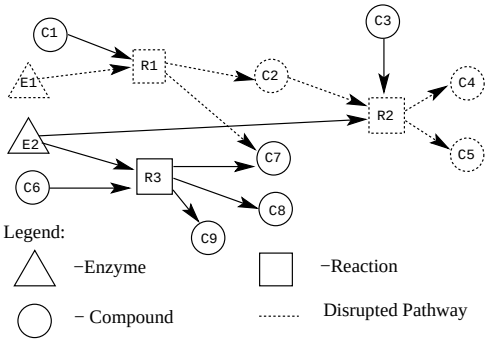

Fig. 1. Graph representation of a metabolic pathway with three reactions R_1, R_2, R_3, two enzymes E_1 and E_2, and nine compounds $C_1, \cdots, C_9$. Dashed lines show the impact of inhibiting enzyme E_1.

There are many models to reconstruct and simulate the metabolism. On such method called *Flux Balance Analysis, (FBA)* [16, 2, 9], computes the flux values of a given pathway at the steady state. *Extreme Pathway Analysis* [19] uses such models to find the path in a pathway that maximizes or minimizes the production of a given compound. This problem is similar to a special case of the enzyme target identification problem considered in this paper. De et. al, for example, consider the extreme pathway analysis problem [5]. In order to reduce the yield of a compound in a pathway, De et. al use FBA to compute the optimal pathway so that the yield of the target metabolite is minimum. They, then, change the concentration of the enzymes in other paths so that these paths are inactive except that optimal one. This method has two major drawbacks. First, it requires changing the concentration for many enzymes. In practice, changing the enzyme concentration is a costly process. Therefore, the number of enzymes whose concentrations are altered should be kept low. Second, the alterations that change the production of a compound can affect the production of other compounds in that pathway. Thus, the solution found by this method can have significant side-effects. In addition to these drawbacks, extreme pathway analysis cannot solve the enzy-

matic target identification problem considered in this paper. Here, we develop methods to overcome the above-mentioned disadvantages and solve a more generic problem.

Contributions: In this paper, we address the enzymatic target identification problem. We overcome the limitations of earlier solutions by allowing fractional values for the quantities of compounds. We develop a measure, named *State-Distance (SD)* that computes the distance between two states of a given pathway. We use SD to measure how much the steady state of a pathway deviates from a given goal state. We develop two algorithms to solve the enzymatic target identification problem based on traversal and genetic algorithms. The former one traverses the search space. Each solution in the search space is a set of enzymes. This algorithm aims to find the enzyme set whose inhibitions have the least SD to the given goal state. Evaluating each potential solution requires computing the steady state of the pathway after inhibiting the enzymes in that solution. To reduce the search cost, it avoids exploring the states that are unlikely to result in a good solution using a filtering and a prioritization strategy. The genetic algorithm uses crossover and mutation to combine existing good solutions to find better ones. It employs the traversal algorithm on small portions of the search space during crossover and a biased randomization technique to improve the quality of the results.

Our experiments using the metabolic pathways from KEGG [14, 15] validate that the computational results of our algorithms agree with the results found by other means in the literature for numerous applications. We observe that our traversal method is a good approximation of the exhaustive search algorithm and it requires significantly less computational time. However, it is only effective for small pathways (i.e., 30-35 enzymes). Our genetic algorithm finds good solutions for large metabolic pathways of size around 100 enzymes in less than 10 minutes.

The rest of the paper is organized as follows. Section 2.1 formally defines the SD measure. Section 2.2 presents the proposed traversal method. Section 2.3 explains the genetic algorithm. Section 3 discusses experimental results. Section 4 concludes the paper.

2 Methods

In this section, we present in silico methods for the enzymatic target identification problem. We first provide a measure to compute the distance between the current steady state and the goal state (Section 2.1). We, then, describe two algorithms, traversal and genetic algorithms (Section 2.2 and 2.3), to solve the enzymatic target identification problem.

2.1 State-Distance

The first task that needs to be addressed is to measure the distance between a given goal state and the steady state of the pathway after inhibiting a set of enzymes. In order to achieve this, we first compute the steady state of the pathway after a set of enzymes are inhibited. We then measure the distance between this state and the goal state. We call this measure the *State-Distance* (SD).

The *state* of a metabolic pathway is a vector that indicates its current status. There are alternative ways to define and compute the state of a given pathway in the literature. These alternatives are similar in spirit. Each entry of the state vector denotes the yield of a compound [27] or a flux in the pathway [17]. Yield of a compound is its amount in the metabolism. The flux of a reaction is the speed at which each compound is produced or consumed by that reaction. We compute the yield of each compound in the steady state using the reaction parameters such as the rate at which it is consumed or produced by each reaction. We apply FBA [16, 2, 9] to get the flux of the pathway in the steady state. Usually, FBA produces a space of steady states that contains infinitely many possible steady states. To select a unique steady state, FBA enforces optimizing an objective function in the solution space. The objective function of FBA often maximizes biomass [6] or the production of ATP [20]. Since the literature contains detailed discussion on the steady state computation, we focus on the distance measure in the rest of this section. We defer the discussion of the steady state computation to the Appendix. In the rest of this paper, each entry of the steady state denotes the yield of a compound unless otherwise stated. Thus, in our notation, the steady state vector for a pathway has as many entries as the number of compounds in that pathway.

Given a goal state for the underlying pathway, next, we discuss how we compute the distance, SD, between the goal state and the current steady state. We, first, present the notation to formally define SD. Assume that the number of compounds in the pathway is m. We denote the goal state as $V_G = [g_1, g_2, \ldots, g_m]$, where g_i = ideal value for the ith entry of the steady state. Let N denote the number of enzymes. The *enzyme vector* shows the inhibition status of the enzymes. We denote it with $V_E = [e_1, e_2, \ldots, e_N]$, where $e_i = \{0, 1\}$ ($e_i = 1$ if E_i is inhibited, otherwise $e_i = 0$). Let $[r_1, r_2, \ldots, r_m]$ be the steady state of that pathway based on the enzyme vector V_E. Let ω_i be a real number that shows the importance of the ith entry of the steady state. We discuss the parameter ω_i later in this section.

We will use the variable d_i to represent the distance contribution of the ith entry of the steady state. We develop two alternative definitions for d_i, namely *exact* and *fuzzy* distance.

– **Exact distance:** This measure is useful when the exact value of the entry in the goal state is known. For the ith entry, we want to approach the goal state as close as possible. We compute the distance as $d_i = \omega_i |g_i - r_i|$.

– **Fuzzy distance:** This measure is useful for extreme pathway analysis or when we do not know the exact values for some entries in the goal state. In this case, we, however, know a lower or upper bound for such entries. In other words, we want to increase (or decrease) the value of that entry to at least (or at most) a given value. Thus, we have two possibilities.
Case 1. (decrease the production of a compound) We want to minimize the ith value of the steady state with a threshold of g_i. In other words, r_i should be smaller than g_i. The smaller the better:

$$d_i = \begin{cases} \infty & \text{if } g_i \leq r_i \\ \omega_i/(g_i - r_i) & \text{for } g_i > r_i \end{cases}$$

Case 2. (increase the production of a compound) We want to maximize the ith value of the steady state with a threshold of g_i. In other words, r_i should be bigger than g_i. The bigger the better.

$$d_i = \begin{cases} \infty & \text{if } g_i \geq r_i \\ \omega_i/(r_i - g_i) & \text{for } g_i < r_i \end{cases}$$

The choice of exact and fuzzy distance depends on the underlying application. Our search algorithm in this paper works for both of them. We use the exact distance measure to compute d_i in the rest of this paper unless otherwise stated.

The weights, ω_i, show the importance of the ith entry of the state vector for the organism. Large ω_i indicates that the ith entry is important. This can be set based on expert input. For simplicity and without loss of generality, we set ω_i = 1 for all i, for the experiment results that are presented in this paper.

We compute SD as the largest of the distance of all the entries of the state vector, i.e., SD = $\max_i\{d_i\} = ||d_i||_1$. Note that one can also define it as SD = $\Sigma_i d_i$. Other combinations of distance measures discussed above are orthogonal with the rest of this paper. Therefore, we do not discuss them further.

Example 1. Consider the pathway in Figure 1. Assume that the goal state is V_G = [0, 0, 1, 0, 0, 0, 1, 1, 1]. That is, we want one unit molecule of each of the compounds C_3, C_7, C_8 and C_9, and none of the remaining compounds. Also, assume that none of the enzymes are inhibited in this pathway (i.e., V_E = [0, 0, 0]). The steady state of this pathway is s_0 = [0, 0, 0.5, 0.5, 0.5, 0, 3, 1, 1] (See Example II.A of the Appendix). The state distance under this condition is $SD(V_E) = ||s_0 - V_G||_1 = 2$.

Now assume that E_1 is inhibited (i.e., V_E = [1, 0, 0]). We compute the steady state after inhibiting E_1 as s_1 = [1, 0, 1, 0, 0, 0, 1, 1, 1] (See Example II.B of the Appendix). Applying the state distance, we get $SD(V_E) = ||s_0 - V_G||_1 = 1$. Thus, inhibiting E_1 brings the steady state closer to the goal state. □

Example 2. Consider the pathway in Figure 1 when no enzyme is inhibited. Assume that the goal state is the same as that in Example 1. With the difference that we want to maximize the yield of the compound C_7 and obtain a yield of at least one unit of it. In this case, we use fuzzy distance for C_7 with a minimum goal = 1. The steady state of the pathway is the same as that in Example 1. Thus, the state distance is $\max\{0, 0, 0.5, 0.5, 0.5, 0, 0.5, 0, 0\} = 0.5$. □

2.2 Traversal Method

Given a metabolic pathway and a goal state, we aim to find the set of enzymes whose inhibitions lead to a steady state with lowest value of SD. One way to solve this problem is to exhaustively examine the SD value after inhibitions of all possible subsets of enzymes. However, this is not feasible because the number of subsets is exponential in the number of enzymes. In this section, we develop a traversal algorithm and two optimization strategies to accelerate it.

We traverse the search space using a branch-and-bound strategy. We consider the search space as a binary tree. Each node of this tree corresponds to a potential solution. Each node records four items; (i) the set of enzymes that are inhibited, (ii) the set of enzymes that are not inhibited, (iii) the set of enzymes that have not been considered so far, and (iv) the SD value of the pathways after all the enzymes in the first set are inhibited. In the root node, the first and the second sets are empty. Therefore, all the enzymes of the pathway are in the third set. We recursively visit the nodes using an in order traversal method [11]. After visiting the current node, we visit the left and right child. Moving from a parent to a child node means considering a new enzyme on top of the enzymes considered in the parent node. The left child denotes that the new enzyme is inhibited and the right child denotes that it is not inhibited. We, then, compute the current SD. If the current SD is less than the best result seen so far, we update the value of best result with the new one. We propose to improve the performance of this algorithm through two different optimizations:

– In many cases, the inhibitions of some uninspected enzymes cannot improve the SD. We set the values in the enzyme vector for such enzymes to zero (i.e. not inhibited). This process, called *Filtering*, can improve the performance of the algorithm as it skips many levels of the search tree.
– The selection of the enzyme when we move from a parent to a child impacts the performance of the algorithm. This is because if the nodes of the top levels in the search tree have small SD, the chance of filtering the nodes in its subtree becomes large. We call this the *Prioritization* strategy.

We discuss these two optimizations later in this section. In order to implement these two optimizations we, first, discuss how we quickly predict SD incrementally when a new enzyme is inhibited.

Predicting the value of SD: Computing the SD value of a given enzyme vector requires computing the steady state of the pathway. An effective prediction strategy can help in avoiding computation of the steady states of a large number of nodes if their SD values are greater than the current best.

We conjecture that the steady state after inhibitions of enzymes E_i and E_j simultaneously is close to the average of that after inhibitions of E_i and E_j separately. This is intuitive, because the influence of inhibitions of two enzymes should be correlated with the total influence of inhibitions of the individual enzymes. We tested this conjecture by randomly sampling 50 enzyme pairs in each of ten randomly selected metabolic pathways of Homo sapiens (*H.sapiens*). The average correlation coefficient [8] of the steady state between actual and predicted values of these 500 random samples was 0.91. This high correlation supports our conjecture.

Filtering strategy : We filter some uninspected nodes as follows. If the predicted SD values of these nodes are bigger than the current minimum SD, we filter these nodes. For a given node in the search tree, let A, B and C denote the set of enzymes that are inhibited, the set of enzymes that are not inhibited and the set of enzymes that are not yet considered respectively. For each enzyme in set C we predict the SD value after that enzyme is inhibited in addition to the enzymes in A. If the predicted value for an enzyme in C is worse than the best SD value found so far, we move that enzyme to set B. Moving h enzymes from set B to set C is equivalent to filtering h levels of the search subtree rooted at the current node. We predict the steady state for a single enzyme during filtering in time proportional to the size of the state vector by precomputing the steady state after inhibition of each enzyme alone.

It is worth noting that we are using each enzyme independently to predict the steady state using a combination of enzymes. This process is error prone and can lead to pruning of potentially useful enzymes when using the filtering strategy. We address this problem by giving each enzyme several chances before we filter it. Let K denote the number of chances, where K is a positive integer. We call this *K-chance strategy*. To incorporate this strategy, we keep a vector, where each entry of this vector denotes the number of times that an enzyme is tested positively for a filter. We filter an enzyme only if that enzyme uses all of its K chances.

Prioritization strategy: We select the most promising enzyme in set C to consider for inhibition. An enzyme is promising if its inhibition in addition to the enzymes in set A has a small SD with high probability. This increases the chance of filtering the nodes in its subtree. Our method works as follows. For each of the uninspected enzymes, we predict the steady state after inhibition of that enzyme in addition to already inhibited enzymes. We then compute the SD between that state and the goal state. We move the enzyme with the smallest predicted SD to set A to create the next child node.

2.3 Genetic Algorithm

The time complexity of the traversal method remains exponential, though the filtering and prioritization strategies reduce the search space significantly. From experiments, the method described in the previous section is only useful for 30-35 enzymes. In this section, we propose a genetic algorithm to solve the target identification problem for large pathways. The genetic algorithm exploits the traversal method as a building block. The main idea of the genetic algorithm is to generate a population of candidate solutions and improve these solutions through crossover and mutation operations. The algorithm stops after a predefined number of *epochs* (or iterations). Next, we describe the genetic algorithm in detail. The genetic algorithm uses the following data structure.

Population: The population P is a set of candidate solutions $\{S_1, S_2, \cdots, S_{num\_seed}\}$. Here, each *solution*, S_i, is a vector that shows which enzymes are inhibited and which enzymes are not. Let N be the number of enzymes in the underlying pathway. We represent a solution with $S_i = [s_1^i, s_2^i, \cdots, s_N^i]$, where s_j^i = 0 means that the enzyme E_j is not inhibited. Similarly, s_j^i = 1 means that E_j is inhibited.

Algorithm 1 summarizes our solution. We discuss the details of this algorithm next.

Algorithm 1 *Genetic Algorithm (Epochs)*

1. Initialize population.
2. **for** $i = 1$ to Epochs **do**
3. Generate children using crossover.
4. Perform selection.
5. Perform mutation.
6. Shrink each solution to minimal subset.
7. **end for**
8. Report the solution with minimal SD.

Step 1. Initialize population. This step generates the initial population, P, which contains candidate solutions. Ideally, a good candidate resembles the optimal solution in terms of both the number and the selection of enzymes that are inhibited by it. However, we do not know the optimal solution at this step. To address this problem, we need to answer two questions. (i) How many enzymes are inhibited in each candidate? (ii) How do we decide which enzymes are inhibited in each candidate? To address the first problem, we employ our traversal algorithm in Section 2.2 as follows. We estimate the number of inhibited enzymes in good solutions (i.e., solutions with low SD). Let λ denote this estimate. We run the traversal method for the top several levels of the search space. We

search 10 levels to limit this traversal time. We, then, select a set of solutions with smallest SD (say 20 best solutions). We estimate λ as the average number of inhibited enzymes in these solutions. The filtering and prioritization strategies push the results with small SD to the top levels of the search tree with high probability. Thus, limiting the search to only a few levels of the tree does not degrade the accuracy of λ greatly.

Once we compute λ, the next problem is to decide which enzymes will be inhibited. We use binomial distribution to compute the inhibition probability of each enzyme. Ideally, the probability of inhibiting an enzyme should be high if that enzyme has a high potential to contribute to a good solution. We predict this potential of each enzyme by considering the SD value obtained after inhibiting that enzyme. Suppose that $SD(E_i)$ denotes the value of SD when only enzyme E_i is inhibited. We conjecture that if $SD(E_i)$ is small, then E_i has a high probability to contribute to a good solution. Let p_i denote the probability of inhibiting enzyme E_i in a good solution. There is a negative correlation between p_i and $SD(E_i)$. Suppose β is the coefficient of that correlation. We use the following equations to estimate p_i.

$$p_i = \beta/(1 + SD(E_i)). \tag{1}$$

This equation requires the value of β. We compute β from the observation that the expected value of the total number of inhibited enzymes is λ. We do this using the following equation.

$$\lambda = \sum_i p_i = \beta \sum_i \frac{1}{1 + SD(E_i)}. \tag{2}$$

From (1) and (2), we compute the probability of inhibiting enzyme E_i as:

$$p_i = \lambda/((1 + SD(E_i)) \sum_i \frac{1}{1 + SD(E_i)}). \tag{3}$$

Now, we are ready to generate candidate solutions. We create each candidate by inhibiting enzyme E_i with probability p_i, $\forall j$. In practice, we generate 100 candidate solutions to create the initial population.

Step 3. Generate children using crossover. This step computes a child population from the current population. It aims to combine two existing solutions to create a better one. Generating a child solution involves the following steps: selection of two parent solutions from the existing population and crossover of these two parents.

We first discuss how we pick two parent solutions from the current population. We conjecture that good parents can generate good children with high probabilities. We, thus, randomly choose each parent using a biased distribution that prefers parents with small SD. Suppose the probability of choosing the solution S_i as a parent is x_i. Based on our conjecture, there is an inverse relation between x_i and $SD(S_i)$. Assume that γ is the coefficient of that correlation. We can write x_i as:

$$x_i = \gamma/(1 + SD(S_i)) \tag{4}$$

We need the parameter γ to compute the probability x_i. We can compute γ from the observation that the selection probabilities of all the solutions in the current population add up to one. Formally, $\sum_i x_i = 1$. Thus, we have

$$1 = \sum x_i = \gamma \sum \frac{1}{1 + SD(S_i)} \tag{5}$$

We get the value of γ from (5) and use it in (4) to compute x_i as:

$$x_i = 1/((1 + SD(S_i)) \sum_i \frac{1}{1 + SD(S_i)}) \tag{6}$$

So far, we have discussed the selection of parents. Next, we discuss the creation of a single child solution from two parent solutions. We denote the parent solutions with F and M ($F, M \in P$), where $F = [f_1, f_2, \cdots, f_N]$ and $M = [m_1, m_2, \cdots, m_N]$. We denote the child of F and M with $Ch = [Ch_1, \cdots, Ch_N]$. We use the following crossover method to produce the child. We first set the enzymes in the child vector for which both parents have the same value. Formally, if $f_j = m_j$, then we set $ch_j = f_j = m_j$. We decide the values of the remaining enzymes using our traversal method in Section 2.2. This strategy favors children that have small SD values as the traversal strategies seeks solutions with small SD.

Table 1 demonstrates this process on an example. In this example, $f_2 = m_2 = 1$, so $ch_2 = 1$. We set the values ch_5, ch_6 and ch_9 similarly. The parents do not agree for the values of ch_1, ch_3, ch_4, ch_7 and ch_8. We use the traversal method to find their values. To do this, we initialize the root of the search tree as follows; the set of inhibited enzymes is $\{E_2, E_5\}$, the set of enzymes that are not inhibited is $\{E_6, E_9\}$, rest of the enzymes are undecided. The traversal algorithm, then, traverses the search space defined by the undecided enzymes to determine their values that minimizes SD.

We repeatedly select two parents and create a child until the number of children is equal to the total number solutions in the initial population.

Table 1. An example showing the generation of a child Ch from two hypothetical parents F and M for a pathway that contains nine enzymes.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
F	0	1	1	0	1	0	0	1	0
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
M	1	1	0	1	1	0	1	0	0
	ch_1	ch_2	ch_3	ch_4	ch_5	ch_6	ch_7	ch_8	ch_9
Ch	?	1	?	?	1	0	?	?	0

Step 4. Perform Selection. At the end of Step 3, we have a set of already existing solutions, P, and a set of child solutions. Thus the total number of solutions is double the size of P (i.e., it is $2{\cdot}num\_seed$). In this step, we update P as the num_seed solutions with the smallest SD among the union of P and the child solutions.

Step 5. Perform mutation. Steps 3 to 4 repeatedly improve the solutions in the initial population P. There is , however, the risk that these steps get stuck in a local minima or a plateau. This step aims to avoid such local minima or plateaus. To do this, we alter the solutions in P by mutation (i.e., changing the inhibition status of some of the enzymes). We mutate each solution in P except the one with the minimum value of SD. For a given mutation rate δ (we used a rate of 0.04), we change the value of each s_j^i to 1 - s_j^i with probability δ.

Step 6. Shrink each solution to minimal subset. In practical applications, solutions that inhibit fewer enzymes are desirable. This is because drugs have to be used to inhibit them. Smaller number of inhibitors are preferable for this purpose as inhibiting an enzyme is a costly task. One way to do this is to inhibit fewer enzymes than given in each solution without changing the SD value of that solution. This step improves the solutions in the population by shrinking the set of inhibited enzymes in each solution as follows. We iteratively test each inhibited enzyme of a solution. For each such enzyme, we update its state to uninhibited. If the SD of the solution does not increase after this modification, we remove this enzyme from the set of inhibited enzymes permanently. Otherwise we keep it. We repeat this iteration until we test all the enzymes.

Use of traversal algorithm in crossover and performance hiccups. The crossover step (Step 3) of our genetic algorithm uses our traversal algorithm to create child solutions with small SD values. We, however, advocated the use of our genetic algorithm over our traversal algorithm for large pathways because the traversal algorithm is not scalable. Thus, we need to show why our genetic algorithm is scalable despite it uses the traversal algorithm for each child at each iteration.

The scalability of the traversal algorithm used in the genetic algorithm follows from the observation that it is used only for the undecided enzymes (i.e., the enzymes for which the two parents disagree). Using the notation we defined in this section, we can compute the probability that enzyme E_i is undecided as $2p_i(1-p_i)$. This is because one parent inhibits E_i while the other does not. Since the inhibition status of each enzyme in a parent is decided independently, the expected number of undecided enzymes is $2\sum_i p_i(1-p_i)$, or simply $2\lambda - 2\sum_i p_i^2$. The actual value of this expectation depends on the probability values p_i. Table 2 lists the expected number of undecided enzymes for four pathways of varying sizes. It shows that the expected value is small, and thus, the traversal algorithm can be used without affecting the scalability of the genetic algorithm in practice.

One can argue that the pathways in Table 2 might have a small expected number of undecided enzymes due to their topology or size, and thus, the search space may be large for larger pathways or pathways with different topologies. To complete our discussion on scalability, we need to compute the expected number of undecided enzymes in the worst possible distribution of probabilities p_i. The following theorem computes the worst expectation and the corresponding standard deviation.

Table 2. The expected number of undecided enzymes for different pathways. #E denotes the number of enzymes.

Metabolic pathway	#E	Exp. num.
Valine, leucine and isoleucine degradation	24	3.40
Glycolysis/Gluconeogenesis	27	2.69
Glycine, serine and threonine metabolism	32	7.15
Purine metabolism	52	1.72

Theorem 1. *Let N denote the number of enzymes in a given pathway. Let λ be the expected number of inhibited enzymes in a solution of the population of solution generated for that pathway by our genetic algorithm. The expected number of undecided enzymes in a child solution is at most $2\lambda - \frac{2\lambda^2}{N}$. The standard deviation for the worst scenario is $\frac{\sqrt{(2\lambda^2 - 2\lambda N + N^2)(N^2-1)N}}{N^2}$.* □

We defer the proof of Theorem 1 to Appendix. Figure 2 plots the expected number of undecided enzymes in the worst case for varying pathway sizes and λ. The vertical lines show the standard deviation in each direction. We plot the expectation for up to $\lambda = 10$, since we observed $\lambda < 10$ in practice. The figure shows that for practical values of λ, the expected number of undecided enzymes remain small enough to make the genetic algorithm efficient. Furthermore, as the pathway size grows by eight fold, the upper bound for the expectation grows by only a few enzymes. Thus, we conclude that our algorithm is scalable to large pathways.

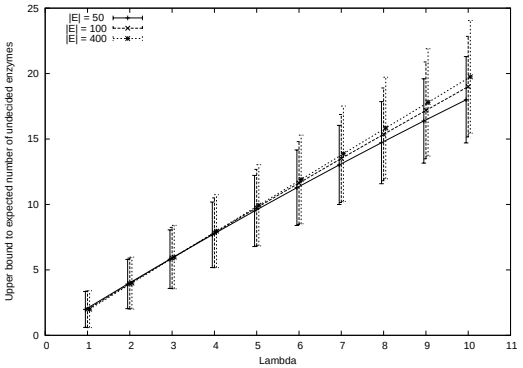

Fig. 2. Upper bound to the expected number of undecided enzymes for different λ and pathway sizes ($N = |E|$). The vertical bars show the standard deviation in each direction.

3 Results

In this section, we evaluate our algorithms on real datasets. We evaluate their biological significance on real applications (Section 3.1). We also evaluate their performance quantitatively (Section 3.2) using the following two measures:

1. *SD:* The SD value is the distance from the goal state. A small SD value indicates a better result.
2. *Execution time:* This indicates the total time (in seconds) taken by our algorithms.

We use the metabolic pathway information of *H.sapiens* and *E.coli* from KEGG as the input dataset (`ftp://ftp.genome.jp/pub/kegg/pathway/organisms/`). We present results for eight pathways of varying sizes in our experiments. Details of these datasets are shown in Table 3.

We implemented the developed algorithms in C++. In our experiments, we set the default values of the parameters as follows. In the computation of SD, ω_i = 1, for all the entries for simplicity. From our experience, the filtering strategy does a good job with two chances. The genetic algorithm works well when the mutation rate, δ, is 0.04. To estimate λ, we search the first 10 levels of the search space to get top 20 best solutions for the balance of efficiency and accuracy. We ran our experiments on a system with dual 2.2 GHz AMD Opteron Processors, 4 gigabytes of RAM, and a Linux operating system.

Table 3. Metabolic pathways from KEGG that are used in our experiments in this paper. KEGG Id is the unique identifier of each pathway in the KEGG database. #E, #R and #C denote the number of enzymes, reactions and compounds respectively in the pathway.

KEGG Id	Metabolic pathway	#E	#R	#C
00980	Metabolism of xenobiotics by cytochrome P450	7	49	57
00670	One carbon pool by folate	17	23	9
00220	Urea cycle and metabolism of amino groups	21	22	27
00310	Lysine degradation	21	25	28
00280	Valine, leucine and isoleucine degradation	24	33	32
00010	Glycolysis/Gluconeogenesis	27	31	25
00260	Glycine, serine and threonine metabolism	32	33	37
00230	Purine metabolism	52	92	65

3.1 Evaluation of the biological significance

We first evaluate the biological significance of our algorithms using flux or yield. We do this by comparing our results to known results in the literature.

Metabolic engineering of Glycolysis/Gluconeogenesis pathway The Glycolysis pathway of *T.pallidum* produces Phosphoenolpyruvate. De et al., [5] studied the path that maximizes the production of this compound. *T.pallidum*, however, has a simple pathway that contains only four enzymes. We considered the Glycolysis/Gluconeogenesis pathway of *E.coli*. This organism has significantly larger and more complex pathway, that contains all the enzymes of *T.pallidum* and many more. We ran our algorithm by setting the goal state to maximization of Phosphoenolpyruvate (i.e., goal > 0 for Phosphoenolpyruvate). Our results suggest inhibiting the set of enzymes {phosphoglucomutase, aldose 1-epimerase , fructose-1,6-bisphosphatase II, phosphoglycerate kinase, pyruvate kinase}, (see red, crossed out enzymes in Figure 6 of the Appendix). When all these enzymes are inhibited, the pathway from alpha-D-Glucose 1-phosphate to Phosphoenolpyruvate is, alpha-D-Glucose 1-phosphate $\rightarrow$ alpha-D-Glucose $\rightarrow$ alpha-D-Glucose 6-phosphate $\leftrightarrows$ (beta-D-Glucose-6P $\leftrightarrows$) beta-D-Fructose-6P $\rightarrow$ beta-D-Fructose-1,6P2 $\leftrightarrows$ (Glycerone-P $\leftrightarrows$) Glyceraldehyde-3P $\leftrightarrows$ Glycerate-1,3P2 $\rightarrow$ Glycerate-3P $\leftrightarrows$ Glycerate-2P $\leftrightarrows$ Phosphoenolpyruvate. This is the same path found for T. pallidum [5]. This means, for the simple life of bacteria, T. pallidum, the main function of Glycolysis pathway is to generate Phosphoenolpyruvate whereas the same pathway for *E.coli* has additional purposes.

Application on the production of Phenylalanine L-Phenylalanine is widely used in the manufacture of aspartame and in parenteral nutrition [1]. Industrial application need an efficient process to generate L-Phenylalanine. Considering the commercial application, Backman et. al selected *E. coli* as the production organism [1]. Thus, we studied the Phenylalanine, tyrosine and tryptophan biosynthesis pathway of *E.coli*. We ran our algorithm by setting the goal state to maximization of Phenylalanine. Our results suggest inhibiting the set of enzymes {phenylalanine translase, chorismate lyase, aspartate aminotransferase, tyrosyl-tRNA synthetase}, (see the red, crossed out enzymes in Figure 7 of the Appendix). Usually, phenylalanine is synthesized from glucose and ammonia in common bacterial systems. Inhibiting phenylalanine translase stops the Phe-tRNA synthesis from phenylalanine. Thus, the concentration of pheylalanine is accumulated. Inhibiting chorismate lyase cuts the flux to other biosynthesis. It, then, reduces the by-products and efficiently uses the raw materials.

Increasing the production of cGMP The compound 3',5'-Cyclic GMP (cGMP) plays an important role in the heart failure. One treatment is to increase cGMP [12, 18, 4]. cGMP appears in the Purine metabolism. We consider this pathway in *H.sapiens* in this experiment. We design the experiment as follows. We first compute the steady state when all the enzymes are active. We consider this steady state as the goal state except the cGMP entry. For the cGMP entry, we set our goal to maximize its production. Our algorithms on the pathway suggests inhibiting the enzymes PDE and guanase. The literature supports this result. PDE inhibitor is a kind of drug that blocks the subtypes of PDE, then it increases the concentration of cAMP or cGMP or both. It can treat heart failure [10, 4]. We predict that the side effect may be less if PDE inhibitor and guanase inhibitor are used together rather than inhibiting PDE alone.

Acetate reduction in E.coli Yang et. al shows several strategies for acetate reduction in the central metabolic pathways of E. coli [28]. One method is to directly influence the formation of acetate. When we run our algorithms on *E.coli* Glycolysis/Gluconeogenesis pathway by setting the goal to minimization of the yield of acetate, the result enzyme is acyl-activating enzyme. This result is consistent with the biological discovery. Inhibition of acyl-activating enzyme cuts down the acetyl-CoA - acetate pathway. The destruction of this pathway results in the low level of acetate [28].

We run our algorithms on the computation of the steady state using flux (details in Appendix). We get another strategy for reduction of Acetate. The results of our algorithms are to inhibit dihydrolipoamide acetyltransferase and pyruvate decarboxylase, which destroy the pathway from Pyruvate to Acetyl-CoA. These results are consistent with the metabolic engineering methods, e.g., the destruction or complete elimination of the portion of the reaction pathway at the acetyl-CoA node [28].

Metabolic engineering of the Butanoate metabolism. Poly-β-hydroxybutyrate is an essential compound for producing plastics. Thus, increasing its production is critical for many industrial applications. Butanoate metabolism produces this compound. We run our algorithm on the Butanoate pathway of *E.coli* with the goal of maximizing this compound. Our results predict that the inhibitions of phosphotransbutyrylase and BHBD increase the entry value of poly-β-hydroxybutyrate while incurring minimum damage to the rest of the metabolism. In fact, Vazque et. al shows the evidence of an association between poly-β-hydroxybutyrate and phosphotransbutyrylase [26].

3.2 Quantitative analysis of the proposed methods

In this section we quantitatively evaluate the performance of our traversal and genetic algorithms.

Evaluation of the traversal method The goal of this experiment is to evaluate the performance of the traversal method. For each pathway, we construct the goal state as the steady state of that pathway when no enzyme is inhibited. We, then, modify the pathway by eliminating an enzyme from that pathway and search the resulting pathway to find the solution that is closest to the goal state. For each pathway, we create one query for each enzyme. Thus, we have as many queries as the number of enzymes. We report the average values over all queries for each pathway.

The experiment described above is based on the following biological intuition. An organism is, often, healthy if all of its enzymes function well. This corresponds to the case when no enzymes are inhibited. Thus, the corresponding steady state is the goal state. An organism may suffer from a disease if an enzyme malfunctions. In this case the query will correspond to a pathway that has malfunctioning enzymes. We aim to find an enzyme set whose inhibition leads it back to the healthy state as close as possible.

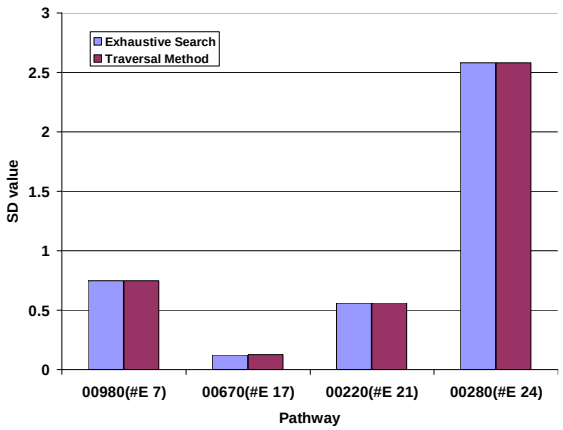

Fig. 3. Average SD values of the traversal method and exhaustive search for different pathways over multiple queries. 'E' followed by a number shows the number of enzymes in the pathway.

Figures 3 and 4 compare traversal method to exhaustive search for pathways that contain up to 24 enzymes. We did not compare the traversal algorithm to exhaustive search for bigger pathways as the exhaustive search requires weeks to months even for a single query. Figure 3 shows the average SD of the solutions computed by the traversal method and those of the exhaustive search algorithm. The results show that the SD values of the traversal method and those of the exhaustive solution are almost identical. Thus, the traversal method is a good approximation to the optimal solution. Figures 4 presents the average running time of the traversal method as compared to the exhaustive search algorithm. The running time of exhaustive search is 1.2 to 11 times that of the traversal method. However, it is also clear that the running time of the traversal method increases exponentially with the number of enzymes. Therefore, it is impractical for very large pathways, although it can be used to solve larger sized problems than the exhaustive search.

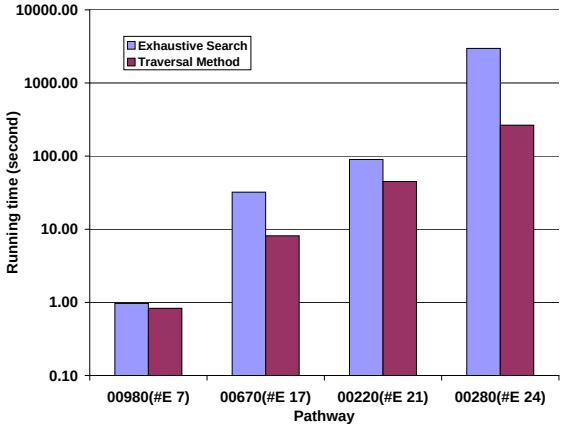

Fig. 4. The average running time (in seconds) of the traversal method and exhaustive search over multiple queries for different pathways. 'E' followed by a number shows the number of enzymes in the pathway.

Evaluation of the genetic algorithm This experiment evaluates the performance of our genetic algorithm. Similar to the evaluation of the traversal method, we design the experiment with pathways up to 84 enzymes. We obtain the pathways that have more than 52 enzymes by combining multiple pathways. For example, Pathway 00230+00790 denotes the pathway obtained by combining 00230 and 00790. For each of these pathways, we created one query for each enzyme similar to the previous section. We report the average values over all queries for each pathway.

The traversal algorithm is impractical for such large pathways due to the exponential time complexity. We, therefore, implement a truncated version of the traversal algorithm. This version truncates all the nodes deeper than L levels, where L is a given parameter. Choosing an appropriate value for L makes the execution time of the truncated method suitable for pathways with a large number of enzymes. For example, the number of enzymes in Glycolysis / Gluconeogenesis pathway (00010) is 27. In the worst case it generates 2^{27} nodes requiring an execution time of 15+ hours. In order to bound the running time of our traversal algorithm for large pathways, we truncated the depth of the search after a fixed depth L. We chose L to be 20 or 23 as the running time quickly becomes impractical for larger L.

Table 4. Comparison of the truncated traversal method (maximum number of levels = 20) and the genetic algorithm for different pathways. D_{20} and T_{20} denote the average SD of the best solution and the average time requirement for multiple queries using the truncated method. D_{ga} and T_{ga} denote the average SD and the average running time for the genetic algorithm respectively. All the time is in seconds. D_0 denotes the average SD between the steady state of the query pathway and the goal state.

	SD	Time
Pathways	$D_0 \mid D_{20} \mid D_{ga}$	$T_{20} \mid T_{ga}$
00280 (#E 24)	2.611 | 2.596 | 2.545	20 | 2
00010 (#E 27)	2.687 | 2.556 | 2.370	166 | 52
00260 (#E 32)	0.837 | 0.819 | 0.797	162 | 38
00230 (#E 52)	10.865 | 1.568 | 1.120	141 | 150
00230+00790 (#E 61)	6.848 | 1.497 | 0.920	58 | 573
00230+00030 (#E 67)	8.590 | 3.682 | 3.159	465 | 410
00230+00340 (#E 67)	5.504 | 1.274 | 0.902	122 | 168
00230+00260 (#E 84)	6.342 | 1.322 | 0.996	121 | 494

Table 4 presents the SD value and the running time of the traversal method and the genetic algorithm for four pathways that have 24 or more enzymes. The difference between the initial damage D_0 and D_{20} shows how much the truncated traversal algorithm reduces the distance between the initial and the goal state. The difference between D_{20} and D_{ga} show the amount of improvement obtained by the genetic algorithm over the truncated traversal method. These results show that the genetic algorithm generates significantly better solutions (lower damage values) as compared to the truncated traversal method for all the cases. The genetic algorithm found solutions that have SD values that are 2% to 39% lower than that found by the truncated traversal algorithm. In addition, the time requirements of the genetic algorithm were comparable or better than the truncated traversal method. For pathway 00010, the genetic algorithm generated on an average 8.06% improvement over the traversal method. The maximum improvement in these experiments was 39%. We have similar comparative gains for other pathways.

One can argue that the effectiveness of the traversal method can be limited due to the limited number of levels. To evaluate the trade off between the accuracy and running time of the truncated algorithm better, we increase the depth of the traversal algorithm until it spends at least as much time as the genetic algorithm for all the pathways. For this purpose, we tested the truncated algorithm for 23 levels. Table 5 shows the comparison between the genetic algorithm and the traversal method with 23 levels for the three largest pathways. From Table 5, the running time of truncated algorithm with the additional 3 levels is up to an order of magnitude higher. However, the accuracy only improved by a small amount and is much lower than the genetic algorithm. It is worth noting that the genetic algorithm required considerably less time for all these cases. Therefore, the genetic algorithm is a good choice for the enzymatic target identification problem for very large pathways.

Table 5. The average SD value and the running time of the truncated traversal algorithm (with maximum number of levels = 23) and the Genetic Algorithm. The running time is reported in seconds.

	SD	Time
Pathways	$D_{23} \mid D_{ga}$	$T_{23} \mid T_{ga}$
00230+00030 (#E 67)	3.552 | 3.159	3310 | 410
00230+00340 (#E 67)	1.242 | 0.902	828 | 168
00230+00260 (#E 84)	1.271 | 0.996	974 | 494

4 Discussion

The goal of enzymatic target identification problem is to identify the set of enzymes whose inhibitions lead to a *steady state* of the metabolic pathway that is close to a goal or desired state. We develop a novel distance measure, State-distance (SD) that measures the damage of the inhibitions of a set of enzymes as a function of the deviation of the entry in the steady state after their inhibitions from that in the goal state. Using this measure, we develop two algorithms that are based on search space traversal and genetic algorithms.

Experiments using the metabolic pathways of *H.sapiens* and *E.coli* from KEGG show that our algorithms can be useful for numerous application including metabolic engineering and biomedicine. Our traversal method is effective for pathways with up to 30-35 enzymes. Our genetic algorithm is effective for arbitrarily large pathways.

References

1. K. Backman, M. J. O'Connor, A. Maruya, E. Rudd, D. McKay, R. Balakrishnan, M. Radjai, V. DiPasquantonio, D. Shoda, and R. Hatch. Genetic engineering of metabolic pathways applied to the production of phenylalanine. *Ann N Y Acad Sci.*, 589, 1990.
2. H. P. J. Bonarius, G. Schmid, and J. Tramper. Flux analysis of underdetermined metabolic networks: The quest for the missing constraints. *Trends Biotechnology*, 15, 1997.
3. G. Q. Chen and Q. Wu. The application of polyhydroxyalkanoates as tissue engineering materials. *Biomaterials*, 26, 2005.
4. H. H. Chen. Heart Failure: A State of Brain Natriuretic Peptide Deficiency or Resistance or Both. *Journal of the American College of Cardiology*, 49(10), 2007.
5. R. K. De, M. Das, and S. Mukhopadhyay. Incorporation of enzyme concentrations into FBA and identification of optimal metabolic pathways. *BMC Systems Biology*, 2(65), 2008.
6. J. S. Edwards and B. O. Palsson. The Escherichia coli MG1655 in silico metabolic genotype: Its definition, characteristics, and capabilities. *Proc Natl Acad Sci U S A*, 97, 2000.
7. J. Fernandes, J. M. Saudubray, G. v. d. Berghe, and J. H. Walter. Inborn Metabolic Diseases: Diagnosis and Treatment. 2006.
8. R. A. Fisher. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10, 1915.
9. J. Forster, I. Famili, P Fu, B. O. Palsson, and J Nielsen. Genome-scale reconstruction of the saccharomyces cerevisiae metabolic network. *Genome Research*, 13, 2003.
10. S. R. Goldsmith. Type 5 Phosphodiesterase Inhibition in Heart Failure: The Next Step. *Journal of the American College of Cardiology*, 50, 2007.
11. E. Horowitz, S. Sahni, and D. Mehta. Fundamentals of data structures in c++. *Silicon Press*, 2007.
12. John C Burnett Jr. Modulating cGMP in heart failure. *BMC Pharmacology*, 7(Suppl 1), 2007.
13. T. Kahveci. Np completeness for optimal enzyme combination identification. Technical report, CISE Department, University of Florida, Jan 2008.
14. M. Kanehisa. A database for post-genome analysis. *Trends in Genetics*, 13(9):375–6, 1997.
15. M. Kanehisa and S. Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 28(1):27–30, Jan 2000.
16. K. J. Kauffman, P. Prakash, and J. S. Edwards. Advances in flux balance analysis. *Current opinion in biotechnology*, 14(5).
17. B. O. Palsson. *Systems biology: Properties of reconstructed networks*. Cambridge University Press, 2006.
18. S. Philipp, J. Monti, I. Pagel, T. Langenickel, T. Notter, F. Ruschitzka, T. Luscher, R. Dietz, and R. Willenbrock. Treatment with darusentan over 21 days improved cGMP generation in patients with chronic heart failure. *Clinical Science*, 103 (Suppl. 48), 2002.
19. N. D. Price, J. L. Reed, J. A. Papin, S. L. Wiback, and B. O. Palsson. Network-based analysis of metabolic regulation in the human red blood cell. *Journal of Theoretical Biology*, 225, 2008.
20. R. Ramakrishna, J. S. Edwards, A. McCulloch, and B. O. Palsson. Fluxbalance analysis of mitochondrial energy metabolism: consequences of systemic stoichiometric constraints. *Am J Physiol Regul Integr Comp Physiol*, 280, 2007.
21. D. Rodriguez-Amaya. Food carotenoids: analysis, composition and alterations during storage and processing of foods. *Forum Nutr*, 56, 2003.
22. C. Scriver, A. L. Beaudet, D. Valle, W. S. Sly, B. Vagelstein, B. Childs, and K. W. Kinzler. *The Online Metabolic and Molecular Bases of Inherited Disease*. New York: McGraw-Hill, 2007.
23. B. Song, P. Sridhar, T. Kahveci, and S. Ranka. Double iterative optimization for metabolic network-based drug target identification. *accepted to International Journal of Data Mining and Bioinformatics*, 2007.
24. P. Sridhar, T. Kahveci, and S. Ranka. An iterative algorithm for metabolic network-based drug target identification. *Pacific Symposium on Biocomputing*, 2007.
25. P. Sridhar, B. Song, T. Kahveci, and S. Ranka. OPMET: A metabolic network-based algorithm for optimal drug target identification. *Pacific Symposium on Biocomputing*, 2008.
26. G. J. Vazque, M. J. Pettinari, and B. S. Mendez. Evidence of an association between poly(3-hydroxybutyrate) accumulation and phosphotransbutyrylase expression in Bacillus megaterium. *Int Microbiol.*, 6(2), 2003.
27. A. I. Vogel, A. R. Tatchell, B. S. Furnis, A. J. Hannaford, and P. W. G. Smith. *Vogel's textbook of practical organic chemistry*. Prentice Hall, 5 edition, 1996.
28. Y. T. Yang, G. N. Bennett, and K. Y. San. Genetic and metabolic engineering. *Electronic Journal of Biotechnology*, 1(3), 1998.

Appendix

The developed algorithms require computing the steady state of the metabolic pathway for each enzyme vector (i.e., candidate solution) that is considered during the search. Since our algorithms evaluate many enzyme vectors, we need to answer the following question: Can we compute the steady state of the pathway efficiently? This question has been studied thoroughly in the literature. Here, we briefly discuss two alternative ways to compute the steady state based on two alternative, yet similar, steady state definitions (Section I and II).

I. Computing the steady state using flux

The flux of a reaction shows the speed at which each compound is produced or consumed by that reaction. As the reactions progress, each flux can increase or decrease other fluxes. The pathway reaches to a steady state when all the flux values remain unchanged over time.

Figure 5 shows a hypothetical pathway along with fluxes that operate on it. Let v_1, v_2, v_3 denote the internal flux for reaction R_1, R_2 and R_3. b_1, b_2, ..., b_8 are the external flux. The state of the pathway shows these current internal and external flux which relates to the other pathways or compounds. Assume the reactions in the pathway are as follows.

$R_1 : C_1 \rightarrow C_2 + 2C_7$

$R_2 : 2C_2 + C_3 \rightarrow C_4 + C_5$

$R_3 : C_6 \rightarrow C_7 + C_8 + C_9$

One way to compute the steady state of this pathway is to employ Flux Balance Analysis (FBA) [16, 2, 9]. FBA creates a matrix, A, that shows how each flux operates on each compound. Each row of this matrix corresponds to a compound and each column corresponds to a flux. For example, for the pathway in Figure 5, A has nine rows and 11 columns since there are nine compounds and 11 fluxes. Consider the flux v_1 which corresponds to reaction R_1 above. Assume that the first column of A corresponds to v_1. The values of A in this column are $A[1,1] = -1$, $A[2,1] = 1$, $A[7,1] = 2$ and all others are zero. This is because v_1 consumes one unit of C_1 to produce one unit of C_2 and two units of C_7. Let v denote the flux vector (i.e., each entry of this vector corresponds to a flux). The product Av, then shows the amount of change in the concentration of each of the compounds. In order to compute the steady state, FBA makes several assumptions. One of them is that the total influx of each compound is equal to the total out flux of that compound. Thus, FBA computes the set of all possible steady states as the solution space of v to the equation

$$\frac{dA}{dt} = Av = 0.$$

Typically, the number of variables in this equation is more than the number of constraints. As a result there are infinite possibilities for x. There are many ways to narrow down the solution space. One of them is that a flux can not be a negative value (i.e., $v_i \geq 0$ for all i). Another assumption is that the maximal rate of incoming flux (the flux enters that the metabolism from external sources) is limited, e.g. ≤ 1. Another way is to include an objective function, such as maximizing the biomass or the production of a specific compound or energy.

Example I.A Consider the pathway in Figure 5. Assume that all the enzymes in this pathway are present in the metabolism and they are not inhibited. Also, assume that as the objective function, we maximize the total output flux. In other words, we want to find the steady state that maximizes $b_4 + b_5 + b_6 + b_7 + b_8$.

Thus, the problem of solving $Av = 0$ translates into the following one: Maximize $b_4 + b_5 + b_6 + b_7 + b_8$ subject to the constraints

- $b_1 - v_1 = 0$
- $v_1 - 2v_2 = 0$
- $b_3 - v_2 = 0$
- $v_2 - b_4 = 0$
- $v_2 - b_5 = 0$
- $b_2 - v_3 = 0$
- $2v_1 + v_3 - b_6 = 0$
- $v_3 - b_7 = 0$
- $v_3 - b_8 = 0$
- $b_1, b_2, b_3 \leq 1$
- $v_1, v_2, v_3, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8 \geq 0$

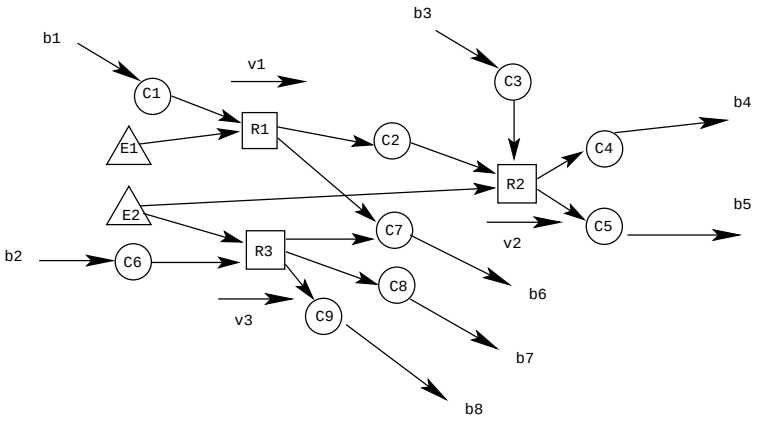

Fig. 5. Flux distribution of a hypothetical pathway.

The solution is, v_1 = 1.0, v_2 = 0.5, v_3 = 1.0, b_1 = 1.0, b_2 = 1.0, b_3 = 0.5, b_4 = 0.5, b_5 = 0.5, b_6 = 3.0, b_7 = 1.0, b_8 = 1.0. We define the steady state as all the flux in the pathway. That is, $[v_1, v_2, v_3, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8]$ = [1, 0.5 1, 1, 1, 0.5, 0.5, 0.5, 3, 1, 1]. □

Example I.B Now assume that E_1 is inhibited in the pathway used in Example I.A, Since E_1 catalyzes the reactionR_1, R_1 becomes inactive. Then v_1 become zero in the flux computation of Example 1. Therefore, we add one more limitation v_1 = 0 to the above flux computation. The steady state under these constraints is [0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1]. □

II. Theoretical yield computation

FBA computes the flux at the steady state. It, however, does not describe the concentration of each compound when the metabolism reaches to a steady state. The amount of each compound at this state is a crucial information for many applications. For example, when dopamine concentration in the brain reduces below a certain level, the motor system nerves become unable to control movement and coordination.

We can compute the amount of each compound in the metabolism as its theoretical yield. The theoretical yield of a compound is the amount of that compound produced by the underlying reactions [27]. Compared to the flux computation, which describes the process of the reactions, the yield computation only depicts the outcomes of the reactions. We can compute the yield of each compound in a pathway for a given initial state using the stoichiometry of that pathway. We use the same matrix A that is used in Section I of the Appendix to denote the stoichiometry. Recall that the stoichiometric coefficients of a reaction show the rate at which it consumes/produces each compound.

In order to compute the yield of each compound in the steady state, we need the initial state of the pathway. For simplicity, we assume that the initial yields of the compounds which are not produced in that pathway are one unit (e.g. one mol). For all other compounds, we set them to zero. Note that the process of computing the steady state is orthogonal to that of the initial state. Thus, one can replace our strategy of selecting the initial state with a different one. We, then, simulate the reactions in the pathway. We assume that all the reactions take place simultaneously given that the enzymes and input compounds required for them are present in the metabolism. For simplicity, we will assume that all reactions take the same amount of time in our discussion. Computing the yield for varying reaction speeds is similar.

Example II.A Assume the reactions in the pathway are the same as that in Section I of the Appendix. Also, assume that all the enzymes are present in the metabolism, similar to Example I.A. In Figure 5, compounds C_1, C_3 and C_6 are not produced in the pathway. They are external inputs to the pathway. Thus, the initial state of the pathway is [1, 0, 1, 0, 0, 1, 0, 0, 0]. In other words, there is one unit of C_1, C_3 and C_6 and none of the other compounds exist initially.

Initially, the reactions, R_1 and R_3, are active. This is because the yields of C_1 and C_6 are non-zero in the initial state. However, the reaction R_2 is inactive because of the lack of C_2. R_1 consumes one mol of C_1, and generates one mol of C_2 and two mol of C_7. R_3 consumes one mol of C_6 and produces one mol of C_7, one mol of C_8 and one mol of C_9. The current state after these reactions take place is $[0, 1, 1, 0, 0, 0, 3, 1, 1]$. At this moment, R_2 is active for both yields of C_2 and C_3 are non-zero. R_1 and R_2 are inactive for the yields of C_1 and C_6 are zero. Then, R_2 consumes one mol of C_2 and $\frac{1}{2}$ mol of C_3, and generates $\frac{1}{2}$ mol of C_4 and $\frac{1}{2}$ mol of C_5. Then, $\frac{1}{2}$ mol of C_3 is left. Thus, the state of the pathway is [0, 0, 0.5, 0.5, 0.5, 0, 3, 1, 1]. Since the yields of C_1, C_2 and C_6 become zero, R_1, R_2 and R_3 are inactive. A steady state is reached at this point as the yield does not change. □

Example II.B Assume that E_1 is inhibited in the pathway used in Example II.A. This makes R_1 inactive as E_1 catalyzes R_1. For consistency, assume the initial state of the pathway is also [1, 0, 1, 0, 0, 1, 0, 0, 0] (i.e., same as that in Example II.A). Then, only R_3 is active for both the input compound C_6 and the enzyme E_3 are present. R_3 consumes one mol of C_6 and generates one mol of C_7, C_8 and C_9. The current state is $[1, 0, 1, 0, 0, 0, 1, 1, 1]$. At this moment, all the reactions become inactive. Therefore, the metabolism comes to a steady state. As compared to the steady state when all the enzymes are present, the yield of C_1 increases and the yield of C_7 decreases. □

There are four important observations that follow from the discussion of steady state for the theoretical yield computation.

- We assume that all reactions take place simultaneously as long as all the input compounds and the enzymes are present in the metabolism. We also assume that each reaction takes place until there are no sufficient compounds that it can consume.
- Theoretically, the state transition of a pathway may lead to an infinite loop of a sequence of more than one state. For example, the translations $a \rightarrow b \rightarrow c \rightarrow a$ show a loop of three states a, b and c. Such loops are considered a sequence of steady states. One can use average, max, min or other functions to summarize such states in a single state. We do not discuss this in detail as it is beyond the scope of this paper.
- The steady state depends on the initial state. One can show that the metabolism is guaranteed to reach to a steady state (or a sequence of steady states) and that the steady state depends on the initial state. The proof of the former follows from the synchronous transition model. The state transition moves each state to a unique next state. Thus, a

transition either creates a new state, remains constant or creates one of the previously visited states. The latter two cases correspond to steady states. The proof of the latter can be done by simply setting the value of C_1 to zero in the initial state of Example II.A. When all the enzymes are present, the steady state after this modification is [0, 0, 1, 0, 0, 0, 1, 1, 1], which is different than the one in that example.

– There is a strong correlation between the steady state of the flux and the steady state of the yield. As the flux reaches to a steady state, the total influx and outflux of each compound remains unchanged. Thus the derivative of the yield of each compound remains unchanged.

The yield computation is a good approximation of the state of the pathway and this value can be measured at lab. It is computationally desirable since it does not require the functional or kinetic information unlike concentration or flux.These information is not available for majority of the existing pathway databases. We use yield to denote state in this paper. Furthermore, this paper is orthogonal to the state computation. Thus, one can easily replace flux or concentration values with yield values.

III. Metabolic engineering of Glycolysis/Gluconeogenesis pathway

Figure 6 shows the Glycolysis/Gluconeogenesis pathway of *E.coli*. We ran our algorithm with the goal of maximizing the production of Phosphoenolpyruvate. Our algorithm predicts that inhibiting the enzymes colored in red and crossed out is the best way to achieve this goal.

IV. Metabolic engineering of Phenylalanine, tyrosine and tryptophan biosynthesis pathway

Figure 7 shows the Phenylalanine, tyrosine and tryptophan biosynthesis pathway of *E.coli*. We ran our algorithm with the goal of maximizing the production of phenylalanine. Our algorithm computes that inhibiting the enzymes colored and crossed out in red is the best way to achieve this goal.

V. Proof of Theorem 1:

In this section, we prove Theorem 1. We denote the expected value of a random variable with E[.]. We will use N to denote the number of enzymes. We will first prove the mean in the first part of the proof. In the second part, we will prove the standard deviation.

Part 1 of proof: The worst scenario analysis of the expected number of undecided enzymes. Let $F = [f_1, f_2, \cdots, f_N]$ and $M = [m_1, m_2, \cdots, m_N]$. denote two arbitrary solutions taken from the population of solutions. In order to compute the expected number of undecided enzymes during the crossover of F and M, we first estimate the number of decided enzymes, that is, the number of enzymes for which, f_i and m_i are the same. We define the random variable X_i as:

$$X_i = \begin{cases} 1 & \text{if } f_i = m_i \\ 0 & \text{otherwise} \end{cases}$$

$X_i = 1$ in two cases: $f_i = m_i = 1$ and $f_i = m_i = 0$. Let p_i be the probability that the ith enzyme in a solution is inhibited. Then, the expected value of X_i is

$$\begin{aligned} \mathrm{E}[X_i] =& p_i \cdot p_i + (1 - p_i) \cdot (1 - p_i) \\ =& 2p_i^2 - 2p_i + 1 \end{aligned}$$

We would like to minimize $E[\sum X_i]$ subject to the constraint that $\sum p_i = \lambda$. The random variables X_i and X_j are independent for $i \neq j$. Therefore, we compute the expected number of decided enzymes as

$$\begin{aligned} \mathrm{E}[\sum_i X_i] &= \sum_i \mathrm{E}[X_i] \\ &= \sum_i (2p_i^2 - 2p_i + 1) \\ &= 2\sum_i p_i^2 - 2\sum_i p_i + N \end{aligned}$$

From the definition of the probabilities p_i in Equation (2) we have $\sum_i p_i = \lambda$.

$$
\begin{aligned}
&\lambda = \sum_i p_i \\
\iff &\frac{\lambda}{N} = \frac{\sum_i p_i}{N} \\
\implies &\frac{\lambda}{N} \leq \sqrt{\frac{\sum_i p_i^2}{N}} \qquad \text{(Follows from power mean inequality)} \\
\implies &\frac{\lambda^2}{N} \leq \sum_i p_i^2
\end{aligned}
$$

Using the inequality we obtained above, we get

$$
\begin{aligned}
\mathrm{E}[\sum_i X_i] &= 2\sum p_i^2 - 2\lambda + N \\
&\geq 2\frac{\lambda^2}{N} - 2\lambda + N
\end{aligned}
$$

By definition of power mean inequality, we minimize the value of the expectation (i.e., equality case) when all $p_i = p_j$ for all i, j. Thus, we conclude that $p_i = \frac{\lambda}{N}$ for all i. Using this value for p_i, we compute the minimum value of $\mathrm{E}[\sum X_i]$ as

$$E[\sum X_i] = \frac{2\lambda^2}{N} - 2\lambda + N.$$

The expected number of undecided enzymes is N - $E[\sum X_i]$. Therefore, the maximum number of undecided enzymes is

$$N - (\frac{2\lambda^2}{N} - 2\lambda + N) = 2\lambda - \frac{2\lambda^2}{N}.$$

Part 2 of proof: The standard deviation of the worst scenario of the expected number of undecided enzymes. Here, we analyze the standard deviation σ for the undecided enzymes in the worst scenario for the mean.

Assume that,

$$X = X_1 + X_2 + \ldots + X_n.$$

From the definition of the standard deviation, σ, is

$$\sigma = \sqrt{\mathrm{E}[(X - \mathrm{E}(X))^2]} = \sqrt{\mathrm{E}[X^2] - (\mathrm{E}[X])^2}.$$

We proved in Part 1 of our proof of Theorem 1 that $\mathrm{E}[X] = \frac{2\lambda^2}{N} - 2\lambda + N$ in the worst scenario, and that this scenario happens when, $p_i = \frac{\lambda}{N}$. Let q be the probability of $X_i = 1$ in the worst scenario. Then,

$$q = p_i \cdot p_i + (1 - p_i) \cdot (1 - p_i) = (\frac{\lambda}{N})^2 + (1 - \frac{\lambda}{N})^2 = 1 - 2 \cdot \frac{\lambda}{N} + 2 \cdot \frac{\lambda^2}{N^2}.$$

Before we compute $\mathrm{E}[X^2]$, we consider $\mathrm{E}[X_i^2]$. The values of X_i is either 0 or 1. Therefore,

$$\mathrm{E}[X_i] = Pr(X_i = 1) = Pr(X_i^2 = 1) = \mathrm{E}[X_i^2].$$

The multiplication $X_i X_j$ $(i \neq j)$ evaluates to one only when $X_i = X_j = 1$. Assume that k of the X_is have value of one and the remaining $N - k$ of the X_is have value of zero. Then, exactly $k(k-1)$ of the $X_i X_j$ multiplications evaluate to one. Let $Z_k = k(k-1)$ denote this. Now, we are ready to consider $\mathrm{E}[X^2]$.

$$
\begin{aligned}
\mathrm{E}[X^2] =& \mathrm{E}\left[(X_1 + X_2 + \ldots + X_n)^2\right] \\
=& \mathrm{E}[\sum_i X_i^2 + \sum_{i \neq j} X_i X_j] \\
=& \mathrm{E}[\sum_i X_i^2] + \mathrm{E}[\sum_{i \neq j} X_i X_j] \\
=& \mathrm{E}[\sum_i X_i] + \mathrm{E}[\sum_{i \neq j} X_i X_j] \quad \text{(by } \mathrm{E}[X_i] = \mathrm{E}[X_i^2]\text{)} \\
=& \mathrm{E}[\sum_i X_i] + \sum_{i \neq j} Pr(X_i = 1, X_j = 1) \\
=& \mathrm{E}[\sum_i X_i] + \sum_{k \geq 2} Pr(X = k) Z_k^2 \\
=& \mathrm{E}[\sum_i X_i] + \sum_{k \geq 2} \binom{N}{k} \cdot q^k (1-q)^{N-k} \cdot Z_k^2 \\
=& \mathrm{E}[\sum_i X_i] + q^2 \sum_{k \geq 2} \left(\frac{N!}{k!(N-k)!} \cdot q^k (1-q)^{N-k} k(k-1) \right) \\
=& \mathrm{E}[\sum_i X_i] + \sum_{k \geq 2} \left(\frac{N!}{(k-2)!(N-k)!} q^k (1-q)^{N-k} \right) \\
=& \mathrm{E}[\sum_i X_i] + N(N-1)q^2 \sum_{k \geq 2} \binom{N-2}{k-2} \cdot q^{k-2} (1-q)^{N-k} \\
=& \mathrm{E}[\sum_i X_i] + N(N-1)q^2 (q + 1 - q)^{N-2} \\
=& \mathrm{E}[\sum_i X_i] + N(N-1)q^2 \\
=& \frac{2\lambda^2}{N} - 2\lambda + N + N(N-1)(1 - 2 \cdot \frac{\lambda}{N} + 2 \cdot \frac{\lambda^2}{N^2})^2 \quad \text{(by } q = 1 - 2 \cdot \frac{\lambda}{N} + 2 \cdot \frac{\lambda^2}{N^2} \text{)}
\end{aligned}
$$

Finally, we compute the standard deviation.

$$
\begin{aligned}
\sigma =& \sqrt{\mathrm{E}[X^2] - (\mathrm{E}[X])^2} \\
=& \sqrt{\frac{2\lambda^2}{N} - 2\lambda + N + N(N-1)(1 - 2 \cdot \frac{\lambda}{N} + 2 \cdot \frac{\lambda^2}{N^2})^2 - (\frac{2\lambda^2}{N} - 2\lambda + N)^2} \\
=& \frac{\sqrt{(2\lambda^2 - 2\lambda N + N^2)(N^2 - 1)N}}{N^2}
\end{aligned}
$$

□

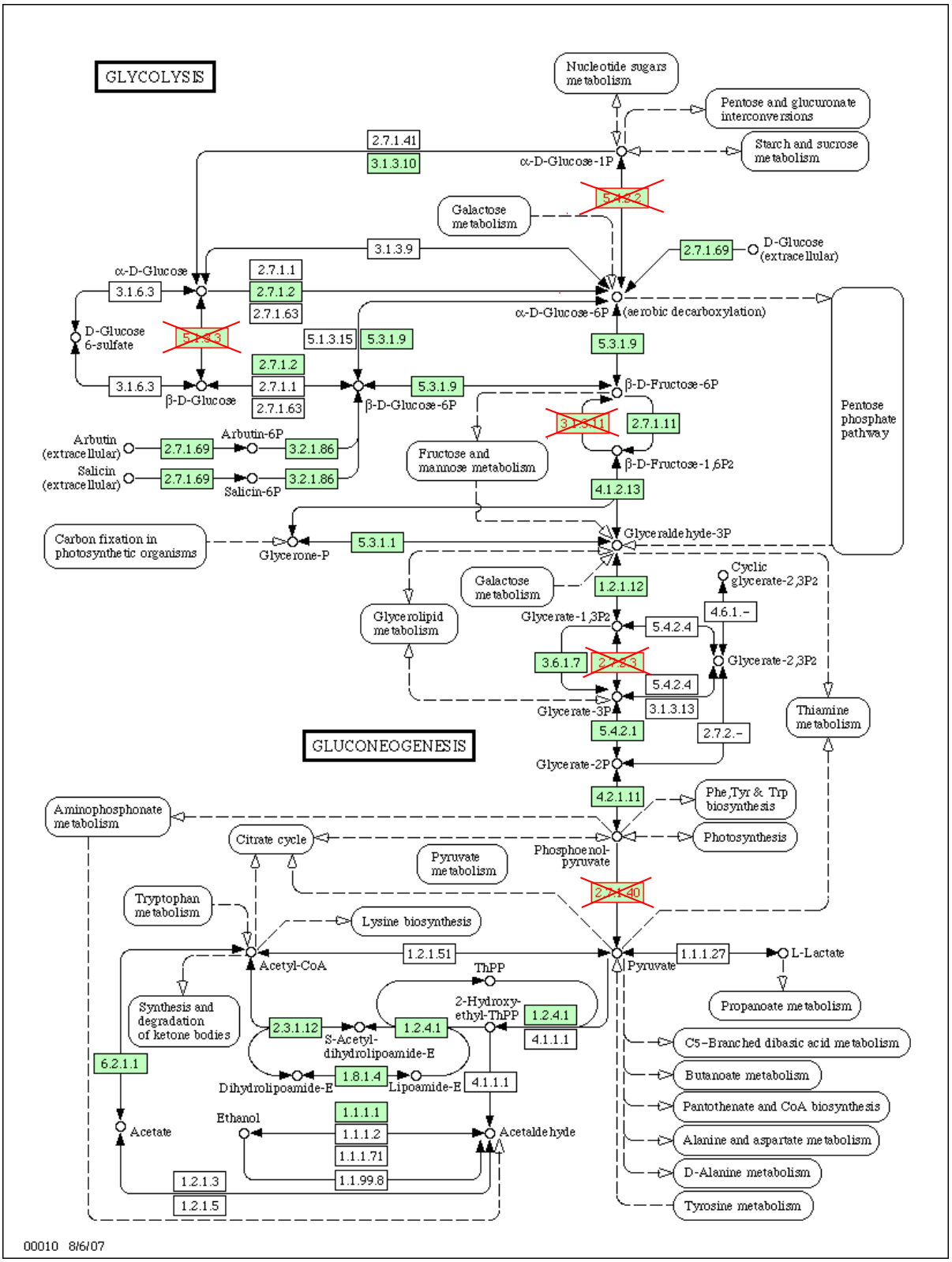

Fig. 6. Glycolysis/Gluconeogenesis for *E.coli*. The enzymes highlighted in green are the enzymes that exist in *E.coli*. According to our algorithm, inhibition of the enzymes marked and crossed out in red maximizes the production of Phosphoenolpyruvate the best.

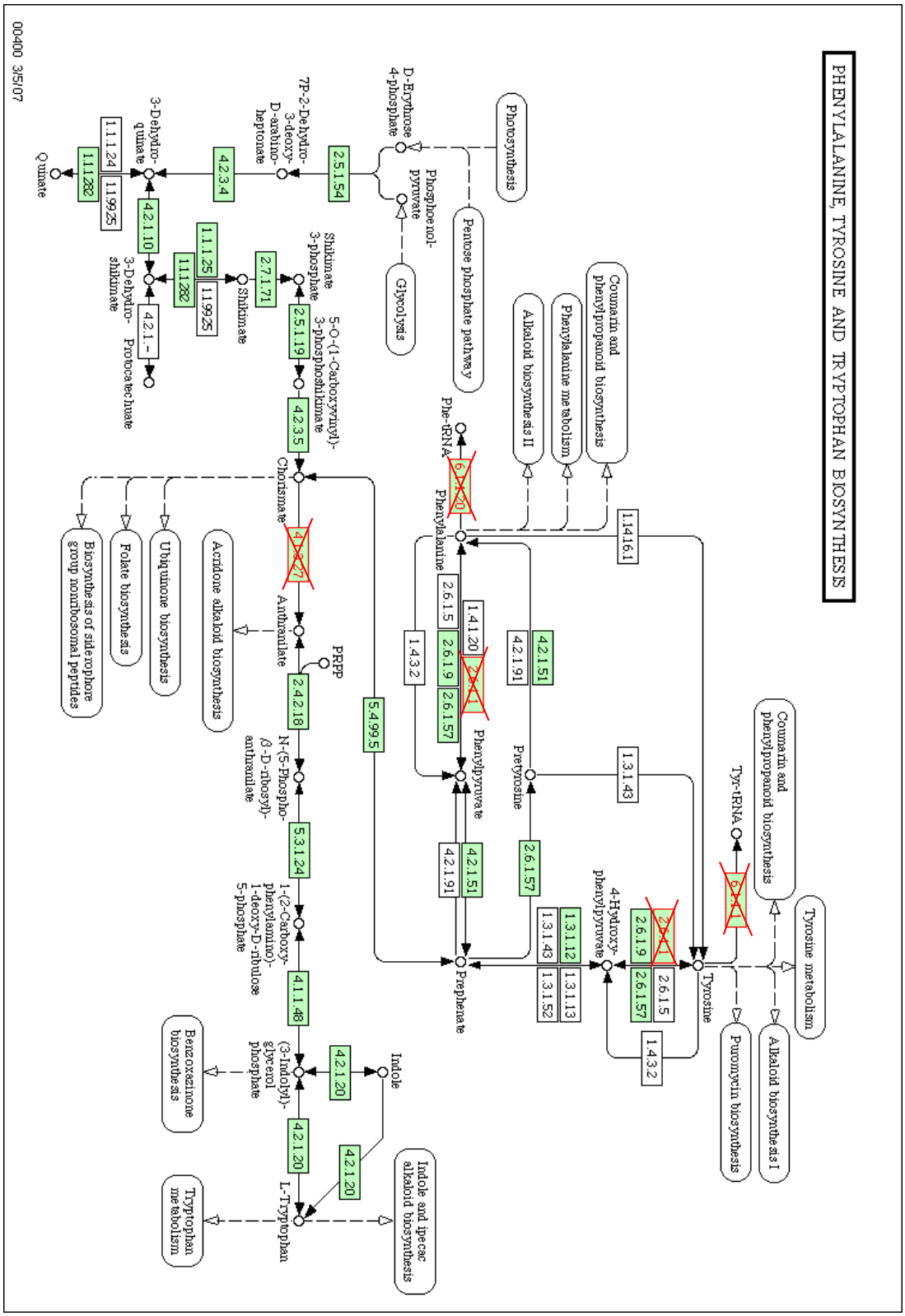

Fig. 7. Phenylalanine, tyrosine and tryptophan biosynthesis for *E.coli*. The enzymes highlighted in green are the enzymes that exist in *E.coli*. According to our algorithm, inhibition of the enzymes marked and crossed out in red maximizes the production of phenylalanine.