Manipulating the Steady State of Metabolic Pathways

Bin Song, İ. Esra Büyüktahtakin, Sanjay Ranka, and Tamer Kahveci

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, biofuels, food industry, and cosmetics. The goal of the enzymatic target identification problem is to identify the set of enzymes whose knockouts lead the metabolism to a state that is close to a given goal state. Given that the size of the search space is exponential in the number of enzymes, the target identification problem is very computationally intensive. We develop efficient algorithms to solve the enzymatic target identification problem in this paper. Unlike existing algorithms, our method works for a broad set of metabolic network models. We measure the effect of the knockouts of a set of enzymes as a function of the deviation of the steady state of the pathway after their knockouts from the goal state. We develop two algorithms to find the enzyme set with minimal deviation from the goal state. 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' results follow those 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, our genetic algorithm can find good solutions in less than 10 minutes.

Index Terms—Metabolic pathway, steady state, traversal approach, genetic algorithm.

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 [35] or the flux [22] in the pathway at a given time. Yield of a compound is the amount of product obtained in the chemical reaction [35]. The flux of a reaction is the rate at which each compound is produced or consumed by that reaction [22]. *Steady state* is the state that remains unchanged over time.

Many applications require altering the steady state of a given pathway. For example, external factors or genetic mutations can change the production rate of a set of enzymes. They can even modify the structure of produced enzymes. 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 [30], [9]. Thus, changing the state of the metabolism back to a desired level is often needed.

Increasing or reducing the production of certain compounds in a metabolism is essential for many industrial applications such as cosmetics and food industry. 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 [4]. Mevalonic acid pathway and MEP/DOXP pathway produce carotenoid that are often used as antioxidant in food industry [28]. The metabolism of many organisms, such as bacteria, algae, and plants, naturally produce these compounds. A common practice is to extract such compounds from these organisms. By manipulating the pathways of these organisms, the production of these compounds can be increased significantly.

One way to change the state of a pathway close to the desired one is to knock out a set of enzymes. When an enzyme is knocked out, 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*, the problem addressed in this paper, aims to identify the set of enzymes whose knockouts lead to a steady state of the metabolic pathway that is as close to a user-supplied goal state as possible. In order to improve the generality of this problem,

• *The authors are with the Department of Computer and Information Science and Engineering, University of Florida, CSE Building, Room E436, Gainesville, FL 32611-6125.*
E-mail: {bsong, ranka, tamer}@cise.ufl.edu, esra@sie.arizona.edu.

Manuscript received 27 Feb. 2009; revised 25 June 2009; accepted 27 Aug. 2009; published online 10 May 2010.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2009-02-0025. Digital Object Identifier no. 10.1109/TCBB.2010.41.

we allow the goal state to be transient as well as steady. In other words, it maybe impossible to reach to the goal state exactly. However, there can still be steady states that are close to the goal state.

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 knocking out a given set of enzymes in polynomial time [36] (see Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.41, for a discussion of the 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 [15]. 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.

There are several algorithms in the literature which aim to address the enzymatic target identification problem. These algorithms, however, make strong assumptions about the metabolism or the problem domain. As a result, they work only for special settings and fail to address this problem when these assumptions do not hold.

1.1 Limitations of Integer Linear Programming-Based Methods

One class of algorithms use integer linear programming to tackle the enzymatic target identification problem. OptKnock is one example to the algorithms in this class [3]. These strategies simulate the metabolism using *Flux Balance Analysis (FBA)* [18], [2], [11]. At a high level, they represent each flux as a variable and solve a linear equation with linear constraints on these variables as follows:

Maximize (or minimize) Objective function/**Subject to** steady state constraints.

This formulation represents the metabolism using a stoichiometric matrix S where the rows and the columns correspond to compounds and fluxes, respectively. Assume that $x = [x_1, x_2, \ldots, x_n]'$ denotes the flux vector for a network with n fluxes. The objective function is typically to maximize a variable or a linear combination of a set of variables. Thus, an objective function is typically $\sum_i c_i x_i$ where c_i are given constants. The constraints define the steady state using stoichiometric model. The solution to the equation $Sx = 0$ is the set of all steady states in this model, and this equation defines the constraints of the integer linear programming problem.

Integer linear programming-based methods have serious drawbacks. First, they are limited to the linear models. In other words, they require that both the objective function and constrains are represented as linear equations (or inequalities). However, many models for metabolic networks do not satisfy this requirement. For example, S-systems [29], [36] defines the steady state as the solution to the equation system

$$\dot{X}_i = \alpha_i \prod_j X_j^{g_{ij}} - \beta_i \prod_j X_j^{h_{ij}} = 0, \quad \forall i.$$

Here, the variable X_i represents the concentration of the ith molecule. $\dot{X}_i$ is the derivative of X_i. The constants α_i, β_i and g_{ij}, h_{ij} denote the rate of the reaction and the rate at which each molecule contributes to a reaction. Clearly, the constraints are nonlinear in the S-systems of equations. Taking the logarithm of the constraints linearizes the constraints as follows: Define $Y_i = \log X_i$. The constraints become

$$\log \alpha_i + \sum_j Y_j g_{ij} - \left(\log \beta_i + \sum_j Y_j h_{ij} \right) = 0, \quad \forall i.$$

However, the objective function transforms into the nonlinear form $\sum_i c_i e^{Y_i}$. Therefore, integer linear programming fails to solve the enzymatic target identification problem.

GMA model [24], [36] is even more problematic for integer linear programming-based methods. This model considers each reaction, that is, a compound is a part of the reaction separately and the model represents the steady state using the following equation:

$$\dot{X}_i = \sum_h \gamma_{ih} \prod_j X_j^{f_{ij}} = 0, \quad \forall i.$$

Here, the constants γ_{ih} and f_{ij} denote the rate of the reaction and the rate at which each molecule contributes to a reaction. In this model, not only the constraints are nonlinear, but also the summation of the multiplicative terms make it impossible to take the logarithm of the constraints.

1.2 Limitations of the Methods for Boolean Network Models

Sridhar et al. [32], [33] and Song et al. [31] considered a simplified version of the enzymatic target identification problem. 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 knockouts eliminate all the *"targeted compounds"* while incurring minimum *"damage."* Targeted compounds are the ones whose productions need to be stopped. They have defined *damage* as the number of nontargeted compounds that are eliminated because of knockouts. Minimum damage is the minimum number of nontargeted compounds eliminated from the metabolism while eliminating the targeted compounds among all possible ways of eliminating the targeted compounds. 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 [33]. Sridhar et al. and Song et al. developed an iterative heuristic method to scale their solutions to large pathways [32], [31]. 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 knockouts of a set of enzymes.

Fig. 1 illustrates the limitations of the binary model used by Sridhar et al. and Song et al. [32], [33]. Inhibiting E_1 knocks out compounds C_2, C_4, and C_5. Suppose C_4 is the

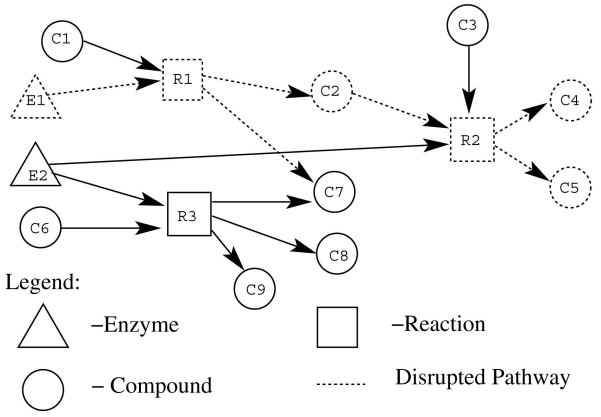

Fig. 1. Graph representation of a metabolic pathway with three reactions R_1, R_2, and R_3, two enzymes E_1 and E_2, and nine compounds $C_1, \ldots, C_9$. Dashed lines show the impact of knocking out enzyme E_1.

targeted compound. Inhibiting E_1 stops two nontargeted compounds (C_2 and C_5). The damage is two using the binary model. There are several drawbacks. One is that the knockout of E_1 accumulates C_1. The binary model, however, ignores this. Furthermore, although the knockout of E_1 does not fully eliminate C_7, it influences the production of C_7. The binary model disregards this influence as well.

Klamt and Gilles introduced a minimal cut set problem, which aimed to find a minimal set of reactions whose deletion leads to no feasible balanced flux distribution in the objective reaction [19]. The authors described several algorithms to solve the minimal cut set problem. However, the minimal cut set model has the drawbacks similar to the one used by Sridhar et al. and Song et al. The aim of this model is to block the objective reaction function which results in the removal of the objective metabolite synthesis. It cannot be used when the aim is to partly decrease or increase the objective metabolites.

1.3 Limitations of Randomized Methods

Extreme Pathway Analysis [26] uses FBA 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 [7]. 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 of 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 enzymatic target identification problem considered in this paper. Here, we develop methods to overcome the above-mentioned disadvantages and solve a more generic problem.

1.4 Limitations of Evolutionary Programming Methods

Patil et al. presented an evolutionary programming method for finding optimal gene deletion strategies [23]. Their method generates a population of random solutions and use genetic algorithm to improve this population. Their method can be applied to nonlinear models as well as linear models. However, it has several drawbacks. The enzymatic target identification problem looks for a set of enzymes that are connected over a complex network and interact through reactions over compounds. Patil et al.'s method ignores these interactions while constructing the population of solutions as well as creating new generation of solutions using crossover. They instead create these solutions randomly. The search space of the enzyme target identification problem is exponential in the number of enzymes. As a result, their method fails to converge to a good solution. Furthermore, the solutions found by their method suggests knocking out unnecessarily large number of enzymes. We experimentally evaluate their method in Section 3.2.3 and defer further discussion to that section.

All of above, we conclude that, existing algorithms work only under a limited specific set of assumptions about the metabolism. Since, these assumptions do not hold in practice most of the time, more generic methods that can address the enzymatic target identification problem for realistic metabolic network models are needed.

1.5 Contributions

In this paper, we address the enzymatic target identification problem. We overcome the limitations of earlier methods, as our algorithm works for a broad set of metabolic network models and literally any objective function. We make a mild assumption that the steady state of the network is computable for the underlying metabolic network model. This assumption hold for a broad set of models.

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 knockouts have the least SD to the given goal state. Evaluating each potential solution requires computing the steady state of the pathway after knocking out 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 genetic algorithm-based method differs from Patil et al.'s [23] from the beginning. It first collects statistics for each enzyme by traversing a small sample of the search space. Using these statistics, it computes the probability for each gene to be knocked out in the best solution. Thus, it creates initial population carefully. Furthermore, it uses

these statistical measures to limit the expected number of enzymes knocked out. During the crossover step, our method uses our traversal algorithm to find the best way of merging two existing solutions. Finally, our method postprocesses each solution produced during the crossover to eliminate redundant enzyme knockouts.

It is worth mentioning that knocking out an essential enzyme can be lethal to the organism. We can avoid choosing such enzymes as candidates for knockouts by simply removing them from the solution space. We however, ignore this step in this paper as it is not the focus of this paper.

Our experiments using the metabolic pathways from KEGG [16], [17] 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 (Sections 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 knocking out a set of enzymes. In order to achieve this, we first compute the steady state of the pathway after a set of enzymes are knocked out. We then measure the distance between this state and the goal state. We call this measure the *State-Distance*.

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 or a flux in the pathway. Yield of a compound is the amount of product obtained in the chemical reaction [35]. The flux of a reaction is the rate at which each compound is produced or consumed by that reaction [22]. 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 [18], [2], [11] or solve S-systems of equations 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 [8] or the production of ATP [27]. Since the literature contains detailed discussion on the steady state computation, we defer the discussion of the steady state computation to Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.41. 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. Note that SD is a generic measure that can be used to represent a broad set of objective functions including the ones used in the literature. 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 knockout 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 knocked out, 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 or topological features of the compounds in the pathway. For simplicity and without loss of generality, we set $\omega_i = 1$ 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., $\mathrm{SD} = \max_i\{d_i\} = \|d_i\|_1$. Note that one can also define it as $\mathrm{SD} = \Sigma_i d_i$. 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 Fig. 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 knocked out 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 B.I of Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.41). The state distance under this condition is $SD(V_E) = \|s_0 - V_G\|_1 = 2$.

Now assume that E_1 is knocked out (i.e., $V_E =$ [1, 0, 0]). We compute the steady state after knocking out E_1 as $s_1 =$ [1, 0, 1, 0, 0, 0, 1, 1, 1] (See Example B.II of Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.41). Applying the state distance, we get $SD(V_E) = \|s_0 - V_G\|_1 = 1$. Thus, knocking out E_1 brings the steady state closer to the goal state.

Example 2. Consider the pathway in Fig. 1 when no enzyme is knocked out. 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 $\mathrm{goal} = 1$. 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 knockouts lead to a steady state with lowest value of SD. One way to solve this problem is to exhaustively examine the SD value after knockouts 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;

1. the set of enzymes that are knocked out,
2. the set of enzymes that are not knocked out,
3. the set of enzymes that have not been considered so far, and
4. the SD value of the pathways after all the enzymes in the first set are knocked out.

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 [13]. 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 knocked out and the right child denotes that it is not knocked out. 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:

- *Optimization 1.* In many cases, the knockouts of some uninspected enzymes cannot improve the SD. We set the values in the enzyme vector for such enzymes to zero (i.e., not knocked out). This process, called *Filtering*, can improve the performance of the algorithm as it skips many levels of the search tree.
- *Optimization 2.* 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 knocked out.

2.2.1 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 knockouts of enzymes E_i and E_j simultaneously is close to the average of that after knockouts of E_i and E_j separately. Note that similar conjectures have been made in the literature [6], [20]. This is intuitive, because the influence of knockouts of two enzymes should be correlated with the total influence of knockouts of the individual enzymes. We tested this conjecture by randomly sampling 50 enzyme pairs in each of 10 randomly selected metabolic pathways of Homo sapiens (*H.sapiens*). The average correlation coefficient [10] of the steady state between actual and predicted values of these 500 random samples was 0.91.

It is worth mentioning that our conjecture above may not hold if E_i and E_j are dependent. The dependency can be in several ways. For example, multiple enzymes may carry out the same reaction. In some cases, the presence of only one of these enzymes suffices for the reaction to occur while in some cases all these enzymes have to be expressed for the reaction to occur. These dependencies may cause over or

underpredictions if the two enzymes create antagonism or synergism, respectively. Then, how much can we trust in this conjecture? In order to answer this question, we performed another experiment as follows: We selected more than 100 pairs of dependent enzymes. Each enzyme pair in this set is within three interactions of each other. We then measure the estimated and actual steady state values after deleting them. The average correlation coefficient between was still 0.91. This high correlation supports our conjecture.

2.2.2 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 knocked out, the set of enzymes that are not knocked out, 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 knocked out 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 knockout 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.

2.2.3 Prioritization Strategy

We select the most promising enzyme in set C to consider for knockout. An enzyme is promising if its knockout 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 knockout of that enzyme in addition to already deleted 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.

2.2.4 Multiple Optimal Solutions

Our traversal algorithm can find multiple solutions (say t solutions) as follows: We store the top t solutions (i.e., the t solutions with smallest SD values) found so far. As we traverse the search space, if we find a solution better than the tth best solution so far, we replace the worst solution in our list with the new one. Also, when filtering the search space, we use the SD value of the tth best solution so far rather than that of the top solution.

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.

2.3.1 Population

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

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

INPUT: *Epochs* $=$ number of epochs the genetic algorithm runs

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 knocked out by it. However, we do not know the optimal solution at this step. To address this problem, we need to answer two questions. 1) How many enzymes are knocked out in each candidate? 2) How do we decide which enzymes are knocked out in each candidate? To address the first problem, we employ our traversal algorithm in Section 2.2 as follows: We estimate the number of removed 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 removed 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 knocked out. We use binomial distribution

to compute the knockout probability of each enzyme. Ideally, the probability of knocking out 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 knocking out that enzyme. Suppose that $SD(E_i)$ denotes the value of SD when only enzyme E_i is knocked out. 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 knocking out enzyme E_i in a good solution. There is a negative correlation between p_i and $SD(E_i)$. We use the following equation to estimate p_i.

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

In this equation, the parameter β is a normalizing constant. We compute the value of β from the observation that the expected value of the total number of removed 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 knocking out enzyme E_i as:

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

Now, we are ready to generate candidate solutions. We create each candidate by knocking out 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

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

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

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

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

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 $F = [f_1, f_2, \ldots, f_N]$ and $M = [m_1, m_2, \ldots, m_N]$. We denote the child of F and M with $Ch = [Ch_1, \ldots, Ch_N]$. 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 removed enzymes is $\{E_2, E_5\}$, the set of enzymes that are not knocked out 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.

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 knockout 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 knock out fewer enzymes are desirable. This is because knocking out an enzyme is a costly task. One way to do this is to knock out fewer enzymes than given in each solution without changing the SD value of that solution. This step improves

TABLE 2
The Expected Number of Undecided Enzymes for Different Pathways

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

#E denotes the number of enzymes.

the solutions in the population by shrinking the set of removed enzymes. In details, we iteratively test each removed enzyme of a solution. For each such enzyme, we update its state to unremoved. If the SD of the solution does not increase after this modification, we remove this enzyme from the set of removed enzymes permanently. Otherwise we keep it. We repeat this iteration until we test all the enzymes. In fact, there can be multiple minimal subsets with the same SD values. We, however, do not see how to compute the SD value of a reduced set of enzymes without actually computing the steady state after reducing the enzyme set. Thus, the number of reduced sets tried while shrinking becomes the bottleneck of this step. Selecting one enzyme at a time for removal guarantees a small upper bound to the running time of the shrinking step. Therefore, we use the above strategy. Note that, one can randomly shuffle the enzymes and try to find alternative reduced sets using different enzymes as seeds. However, to keep the focus of the paper, we do not provide experiments with alternative implementations of the shrinking step.

Finding multiple solutions. An obvious question that follows the genetic algorithm is whether it can find multiple alternative solutions rather than a single solution. Our genetic algorithm can return alternative solutions in two different ways. First, recall that each generation contains a population of solutions. Thus, each member of each population is a solution itself. Keeping the top t solutions with smallest SD values will return t alternative solutions. Second, the shrinking phase of our genetic algorithm can be altered to return alternative shrink sets of enzymes with the same SD values. This can be done by shrinking the enzymes in random order multiple times as discussed above. Each random ordering can produce another alternative minimal subset of enzymes that has the same SD value if such alternative minimal subsets exist.

2.4 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 knocks out E_i while the other does not. Since the knockout 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.

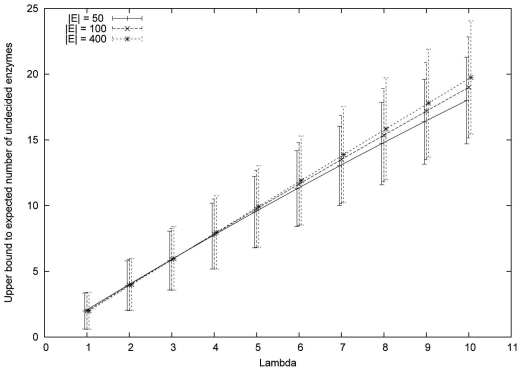

Fig. 2. Upper bound to the expected number of undecided enzymes for different λ and pathway sizes ($N = |E|$). The vertical bars show the standard deviation in each direction.

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 maybe 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.

Theorem 1. *Let N denote the number of enzymes in a given pathway. Let λ be the expected number of removed 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*

$$\frac{\sqrt{(2\lambda^2 - 2\lambda N + N^2)(N^2 - 1)N}}{N^2}.$$

Proof. See Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.41. □

Fig. 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 eightfold, the upper bound for the expectation grows by only a few enzymes. *Thus, we conclude that our algorithm is scalable to large pathways.*

TABLE 3
Metabolic Pathways from KEGG that are Used in Our Experiments in This Paper

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
-	Energy Metabolism	50	46	60
-	Amino Acid Metabolism	195	317	305
-	whole metabolism	640	1176	1067

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. The symbol "-" means that no KEGG Id exists for this metabolism.

3 Results

In this section, we evaluate our algorithms on real data sets. 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 data set. We present results for eight pathways of varying sizes in our experiments. Details of these data sets 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, $\omega_i = 1$, 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.

We use the following strategy to set the initial state of the metabolic networks in our experiments. Some of the compounds used in the metabolic pathway are produced within the same pathway while some are supplied from external sources. This information is available in the existing metabolic pathway databases. In our experiments, we assume the same amount (one mol) of each compound that is externally supplied. This is because we do not want to artificially put any bias toward any externally supplied compound. We also assume that the internally produced compounds initially do not exist in the system. This is because, it is realistic to let the reactions decide which one is produced more rather than artificially setting some values. That said, it is important to realize that our algorithms can work for any initial state as they do not benefit from these assumptions.

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.

3.1.1 Metabolic Engineering of Glycolysis/Gluconeogenesis Pathway

The Glycolysis pathway of *T.pallidum* produces Phosphoenolpyruvate. De et al. [7] 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 knocking out the set of enzymes {phosphoglucomutase, aldose 1-epimerase, fructose-1,6-bisphosphatase II, phosphoglycerate kinase, pyruvate kinase}, (see red, crossed out enzymes in Fig. 3). When all these enzymes are knocked out, 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 $\leftrightarrow$ (beta-D-Glucose-6P $\leftrightarrow$) beta-D-Fructose-6P $\rightarrow$ beta-D-Fructose-1,6P2 $\leftrightarrow$ (Glycerone-P $\leftrightarrow$) Glyceraldehyde-3P $\leftrightarrow$ Glycerate-1,3P2 $\rightarrow$ Glycerate-3P $\leftrightarrow$ Glycerate-2P $\leftrightarrow$ Phosphoenolpyruvate. This is the same path found for T. pallidum [7]. This can have several explanations. The enzyme fructose-bisphosphatase takes part in several pathways of *E. coli* such as the Fructose and mannose metabolism. This enzyme, however, slows down the production of beta-D-Fructose-1,6P2 which is needed for production of Phosphoenolpyruvate. Our results suggest that knocking this enzyme out will accelerate the production of Phosphoenolpyruvate. This enzyme, however, does not appear in *T. pallidum*. Therefore, the

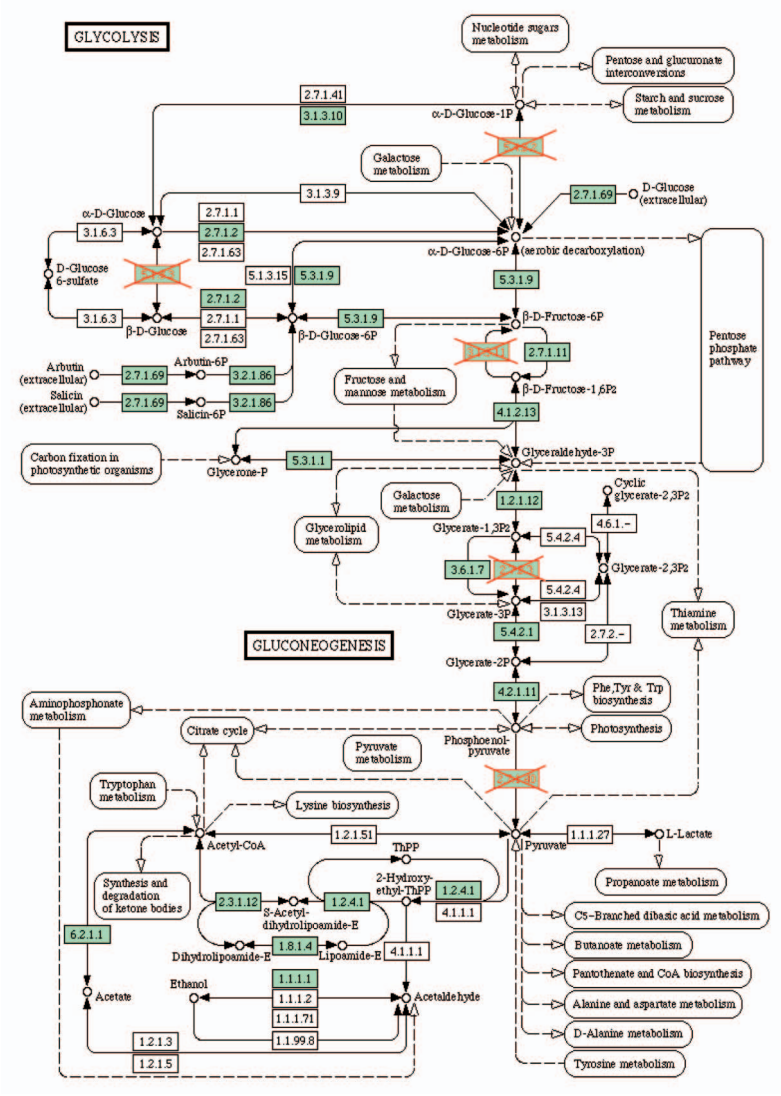

Fig. 3. Glycolysis/Gluconeogenesis for *E.coli*. The enzymes highlighted in green are the enzymes that exist in *E.coli*. According to our algorithm, knockout of the enzymes marked and crossed out in red maximizes the production of Phosphoenolpyruvate the best.

production of Phosphoenolpyruvate goes through the same path in *T. pallidum* without interruption of other enzymes. Similarly, pyruvate kinase converts Phosphoenolpyruvate to pyruvate in *E.coli*, but not in *T. pallidum*. Thus, it reduces the amount of Phosphoenolpyruvate in *E.coli*. This example shows that we can successfully identify these factors and optimize the production of the desired compounds.

3.1.2 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 knocking out the set of enzymes {phenylalanine translase, chorismate lyase, aspartate aminotransferase, tyrosyl-tRNA synthetase}, (see the red, crossed out enzymes in Fig. 4). 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.

3.1.3 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 [14], [25], [5]. 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 suggest knocking out 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

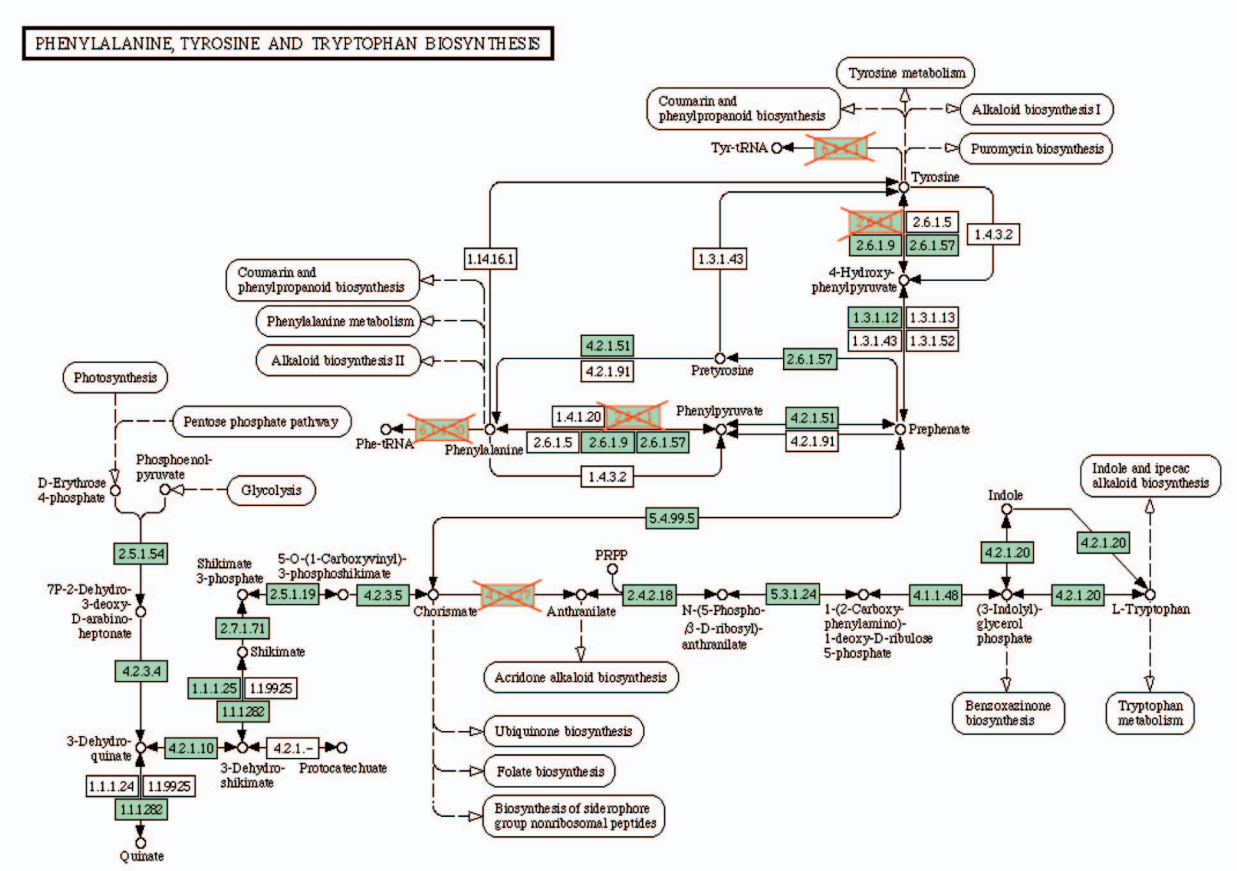

Fig. 4. Phenylalanine, tyrosine, and tryptophan biosynthesis for *E.coli*. The enzymes highlighted in green are the enzymes that exist in *E.coli*. According to our algorithm, knockout of the enzymes marked and crossed out in red maximizes the production of phenylalanine.

failure [12], [5]. We predict that the side effect maybe less if PDE inhibitor and guanase inhibitor are used together rather than knocking out PDE alone.

3.1.4 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 knockouts 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 [34].

3.1.5 Acetate Reduction in E.coli

In this experiment, we show how our method performs when it is applied on a metabolism using the linear constraints of the classical flux balance analysis.

Yang et al. shows several strategies for acetate reduction in the central metabolic pathways of E. coli [38]. 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 [38]. The results found by our algorithms suggest to knockout 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 [38].

3.1.6 Purine Metabolism on Generalized Mass Action (GMA) Model

In this example, we demonstrate the results found by our method when it is applied on a metabolism using a nonlinear constraints. One of the significant productions in Purine metabolism is Uric acid. When the concentration of Uric acid in human body is above the normal range, it results in some diseases, such as, hyperuricemia and gout. We apply our algorithms to Purine metabolism on GMA model. The GMA model considers the concentration of each compound and it is nonlinear system (see Section 1). We utilize the GMA model on Purine metabolism in Chapter 10 in Voit's book [37]. We run our genetic algorithm on this metabolism by setting the goal to minimize the concentration of uric acid and keep the concentration of other compounds unchanged. The resulting enzyme set is {xanthine dehydrogenase, adenosine deaminase, Oxidoreductases}. From the literature, we know that inhibiting xanthine dehydrogenase or oxidase is widely used for

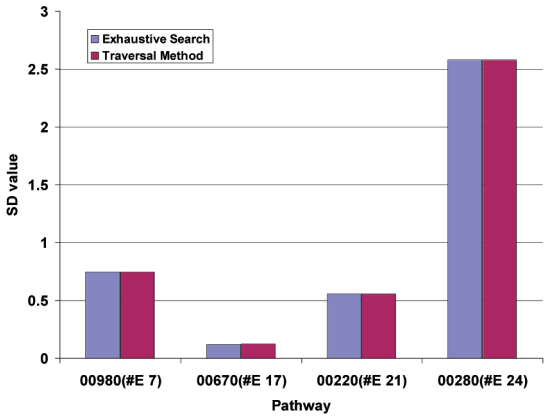

Fig. 5. Average SD values of the traversal method and exhaustive search for different pathways over multiple queries. "E" followed by a number shows the number of enzymes in the pathway.

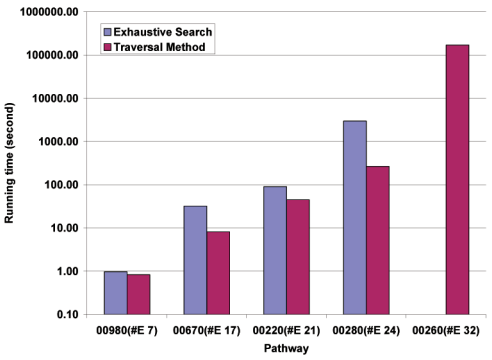

Fig. 6. The average running time (in seconds) of the traversal method and exhaustive search over multiple queries for different pathways. "E" followed by a number shows the number of enzymes in the pathway.

treatment of hyperuricemia and gout [21]. When uric acid production is blocked by inhibiting of xanthine oxidase, it results in an increase in hypoxanthine and xanthine. Thus, inhibiting adenosine deaminase can decease the concentration of hypoxanthine and xanthine.

3.2 Quantitative Analysis of the Proposed Methods

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

3.2.1 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 knocked out. 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 knocked out. 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 knockout leads it back to the healthy state as close as possible.

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. For example, the exhaustive method will cost days when the number of enzymes are more than 30. Fig. 5 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. Fig. 6 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. For example, in Glycine, serine, and threonine metabolism (the number of enzymes is 32), the running time of the traversal method is more than two days. Therefore, it is impractical for very large pathways, although it can be used to solve larger sized problems than the exhaustive search.

Our final experiment, in this section, evaluates the variation of the number of enzymes knocked out in the top results found by our traversal algorithm. We measure this as follows: We use the Urea cycle and metabolism of amino groups in this experiment. We run a query after deleting each of the 21 enzymes in this network. We find the top 10 solutions for each of the 21 queries (i.e., totally 210 solutions). Fig. 7 shows the distribution of the number of enzymes knocked out in these solutions. The results show that the number of enzymes deleted can vary. However, they cluster around several values (in this case, two values). This resembles a mixed Gaussian distribution.

3.2.2 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

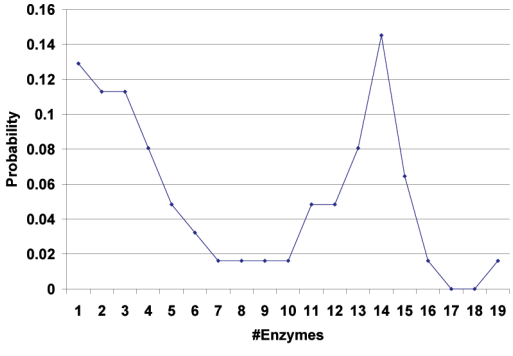

Fig. 7. Distribution of the number of enzymes in the solutions for Urea cycle and metabolism of amino groups.

TABLE 4
Comparison of the Truncated Traversal Method ($\text{Maximum Number of Levels} = 20$) and the Genetic Algorithm for Different Pathways

	SD				Time		
Pathways	D_0	D_{20}	D_{tral}	D_{ga}	T_{20}	T_{tral}	T_{ga}
00980 (#E 7)	0.808	0.75	0.75	0.75	0.83	0.83	1
00220 (#E 21)	0.970	0.882	0.53	0.88	28	40	2
00280 (#E 24)	2.611	2.596	2.581	2.581	20	265	2
00010 (#E 27)	2.687	2.556	2.454	2.370	166	8386	52
00260 (#E 32)	0.837	0.819	?	0.797	162	∞	38
00230 (#E 52)	11.218	1.160	?	1.156	145	∞	153
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

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_{tral} and T_{tral} denote the average SD and the average running time for the traversal 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. ∞ denotes that the running time is more than one day. ? denotes that no SD values can be computed within one day.

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 presents the SD value and the running time of the traversal method and the genetic algorithm for different pathways. Fig. 5 had already demonstrated that our traversal algorithm finds near optimal solutions for small sized pathways. This results in this table show that the performance of the genetic algorithm is comparable to the traversal method. Thus, it finds accurate results. For example, in Metabolism of xenobiotics by cytochrome P450 (00980), SD values and running time are similar for both methods. In Urea cycle and metabolism of amino groups (00220), the genetic algorithm runs much faster while it obtains worse SD values than the traversal method. When the size of pathways increases, the performance of the genetic algorithm is much better than the traversal method.

In Table 4, 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 percent 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 percent improvement over the traversal method. The maximum improvement in these experiments was 39 percent. 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 largest pathways. From Table 5, the running time of truncated algorithm with the additional three 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 most of these cases. Therefore, the genetic algorithm is a good choice for the enzymatic target identification problem for very large pathways.

We apply our genetic algorithm to the large metabolism. For the energy metabolism and amino acid metabolism, our genetic algorithm costs less than one hour. Even for the whole metabolism, our genetic algorithm runs less than 12 hours. Details are in Table 6.

TABLE 5
The Average SD Value and the Running Time of the Truncated Traversal Algorithm (with $\text{Maximum Number of Levels} = 23$) and the Genetic Algorithm

	SD	Time
Pathways	$D_{23} \mid D_{ga}$	$T_{23} \mid T_{ga}$
00280 (#E 24)	2.581 | 2.581	144 | 2
00010 (#E 27)	2.525 | 2.370	987 | 52
00260 (#E 32)	0.819 | 0.797	1042 | 38
00230 (#E 52)	1.568 | 1.084	1366 | 153
00230+00790 (#E 61)	1.368 | 0.920	335 | 573
00230+00030 (#E 67)	3.552 | 3.159	3310 | 410
00230+00340 (#E 67)	1.242 | 1.064	828 | 168
00230+00260 (#E 84)	1.271 | 0.996	974 | 494

The running time is reported in seconds.

In genetic algorithm, there exist several parameters. First, we test the impact of mutation rate on the SD values. Table 7 presents the average SD values with different mutation rates in several pathways. In Table 7, the average SD values do not vary much with different mutation rate. Thus, we select $\delta = 0.04$ as we observed slightly better SD values with this choice in our experiments.

In Step 1 of the genetic algorithm, to estimate λ, we search the first 10 levels of the search space to get top 20 best solutions. Table 8 presents the estimated λ values and the cost time for different levels for the Glycolysis/Gluconeogenesis pathways. As we increase the number of levels, the value of λ increases monotonically. This is because each level suggests knocking out new enzymes on top of already knocked out ones. However, although the number of solutions increase exponentially as we look at deeper levels, the estimated value of λ grows slowly. Between levels 8 and 10, we estimate the actual value of λ correctly. Table 8 also shows that traversing more levels increases the running time exponentially. In practice, we have observed that there is no improvement in the SD value found by our genetic algorithm as we go traverse more levels to estimate λ. Also the running time grows exponentially with the number of levels. Therefore, in order to balance efficiency and accuracy, we traverse only 10 levels for these practical purposes.

3.2.3 Comparison to an Existing Genetic Algorithm Method

Our last experiment compares the performance of our genetic algorithm to a recent work. Patil et al. presented an evolutionary programming method for finding optimal gene deletion strategies [23]. Their method uses genetic algorithm to generate a population of random solutions. This method can be applied to our enzymatic target identification problem directly. We implement Patil's method in C++. We compare our genetic algorithm with Patil's method in accuracy. Table 9 shows the SD value of Patil's method and our genetic algorithm for seven pathways and large metabolism.

Our method consistently outperforms Patil et al.'s method in all test cases. The SD value of Patil et al.'s method is 40 to 3,000 percent more than that of our method. One obvious question is whether our method finds better SD values at the expense of knocking out more enzymes than Patil et al.'s method. The last two columns of Table 9 show that Patil et al.'s method is knocking out up to 20.9 times more enzymes than our algorithm on the average. We conclude that our algorithm is superior both in terms of finding a solution close to the goal state and the cost in doing that.

The reasons behind the success of our method over Patil et al.'s is that our method randomly knocks out each enzyme using a different probability distribution. This distribution depends on the likelihood that the each enzyme is a part of a good solution. Furthermore, unlike Patil et al.'s method, our crossover strategy optimizes the child solution created at each generation.

TABLE 6
The Average Running Time of Our Genetic Algorithm for Large Metabolism

Metabolism	Time
Energy Metabolism (#E 50)	90
Amino Acid Metabolism (#E 195)	2794
whole metabolism (#E 640)	43262

The running time is reported in seconds.

TABLE 7
The Average SD Values with Different Mutation Rates δ in Several Pathways

Pathways	$\delta = 0.02$	$\delta = 0.04$	$\delta = 0.06$
00010	2.370	2.370	2.370
00230	1.131	1.084	1.102

4 Discussion

The goal of enzymatic target identification problem is to identify the set of enzymes whose knockouts lead to a *steady state* of the metabolic pathway that is close to a goal or desired state. We develop a novel distance measure, SD that measures the damage of the knockouts of a set of enzymes as a function of the deviation of the entry in the steady state

TABLE 8
The Estimated λ Values and the Cost Time for Different Levels for the Glycolysis/Gluconeogenesis Pathway

levels	estimated λ value	time(sec)
6	1.55	19
8	2.19	21
10	2.52	29
12	3.08	64
14	3.53	119

The average number of knocked out enzymes in the optimal solution of this pathway is 2.2.

TABLE 9
The Average SD Value and the Running Time of Patil's Method and Our Genetic Algorithm

	SD	NE
Pathways	$D_{Pat} \mid D_{ga}$	$NE_{Pat} \mid NE_{ga}$
00280 (#E 24)	3.572 | 2.545	9.18 | 1.59
00010 (#E 27)	4.207 | 2.370	10.59 | 2.19
00260 (#E 32)	1.039 | 0.797	16.22 | 1.21
00230 (#E 52)	5.968 | 1.084	19.57 | 4.47
00230+00030 (#E 67)	13.996 | 3.159	28.47 | 3.83
00230+00340 (#E 67)	4.698 | 1.064	28.31 | 3.59
00230+00260 (#E 84)	5.754 | 0.996	35.72 | 3.27
Amino Acid Metabolism (#E 195)	2.408 | 0.737	95.1 | 1.3
whole metabolism (#E 640)	60.904 | 1.962	315.5 | 5.25

The running time is reported in seconds. D_{Pat} and D_{ga} denote the average SD for Patil's and our method, respectively. Similarly, NE_{Pat} and NE_{ga} denote the average number of enzymes knocked out using Patil's method and our genetic algorithm, respectively.

after their knockouts 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 applications 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.

In Section 2, we discuss the enzyme deletion strategies for the enzymatic target identification problem. However, genetic manipulations can lead to partial reduction of the enzyme concentrations rather than knocking it out entirely. One way to deal with this is to represent an enzyme status by an integer variable, $\{0, 1, 2, \ldots, m\}$ that shows the level of enzyme activity. Here, the enzyme can exist in $m+1$ status. 0 means that the enzyme is inactive. 1 to m presents the activity level of the enzyme with m being the highest activity. In this situation, our methods, traversal method and genetic algorithm can still work. Genetic algorithm can be used to this situation directly. For traversal method, the search space is no longer a binary tree. For each node, it has $m+1$ children which denotes the $m+1$ enzyme status. The filtering and prioritization strategy can also be applied to this situation. We expect that this will hurt the running time of the traversal method significantly.

ACKNOWLEDGMENTS

This work was supported partially by the US National Science Foundation (NSF) under grants CCF-0829867 and IIS-0845439, and UF Research Initiatives grant (00072365).

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," *Annals of the New York Academy of Sciences,* vol. 589, pp. 16-24, 1990.

[2] H.P.J. Bonarius, G. Schmid, and J. Tramper, "Flux Analysis of Underdetermined Metabolic Networks: The Quest for the Missing Constraints," *Trends in Biotechnology,* vol. 15, pp. 308-314, 1997.

[3] A.P. Burgard, P. Pharkya, and C.D. Maranas, "Optknock: A Bilevel Programming Framework for Identifying Gene Knockout Strategies for Microbial Strain Optimization," *Biotechnology and Bioeng.,* vol. 84, pp. 647-657, 2003.

[4] G.Q. Chen and Q. Wu, "The Application of Polyhydroxyalkanoates as Tissue Engineering Materials," *Biomaterials,* vol. 26, pp. 6565-6578, 2005.

[5] H.H. Chen, "Heart Failure: A State of Brain Natriuretic Peptide Deficiency or Resistance or Both," *J. Am. College of Cardiology,* vol. 49, no. 10, pp. 1089-1091, 2007.

[6] T.C. Chou, "Theoretical Basis, Experimental Design, and Computerized Simulation of Synergism and Antagonism in Drug Combination Studies," *Pharmacological Rev.,* vol. 58, no. 3, pp. 621-681, 2006.

[7] R.K. De, M. Das, and S. Mukhopadhyay, "Incorporation of Enzyme Concentrations into FBA and Identification of Optimal Metabolic Pathways," *BMC Systems Biology,* vol. 2, no. 65, 2008.

[8] J.S. Edwards and B.O. Palsson, "The Escherichia coli MG1655 in Silico Metabolic Genotype: Its Definition, Characteristics, and Capabilities," *Proc. Nat'l Academy of Sciences USA,* vol. 97, pp. 5528-5533, 2000.

[9] J. Fernandes, J.M. Saudubray, G.v.d. Berghe, and J.H. Walter, *Inborn Metabolic Diseases: Diagnosis and Treatment.* Springer, 2006.

[10] R.A. Fisher, "Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population," *Biometrika,* vol. 10, pp. 507-521, 1915.

[11] J. Forster, I. Famili, P. Fu, B.O. Palsson, and J. Nielsen, "Genome-Scale Reconstruction of the Saccharomyces Cerevisiae Metabolic Network," *Genome Research,* vol. 13, pp. 244-253, 2003.

[12] S.R. Goldsmith, "Type 5 Phosphodiesterase Inhibition in Heart Failure: The Next Step," *J. Am. College of Cardiology,* vol. 50, pp. 2145-2147, 2007.

[13] E. Horowitz, S. Sahni, and D. Mehta, "Fundamentals of Data Structures in C++," *Silicon Press,* 2007.

[14] J.C. Burnett Jr., "Modulating cGMP in Heart Failure," *BMC Pharmacology,* vol. 7, no. S14, 2007.

[15] T. Kahveci, "Np Completeness for Optimal Enzyme Combination Identification," technical report, CISE Dept., Univ. of Florida, Jan. 2008.

[16] M. Kanehisa, "A Database for Post-Genome Analysis," *Trends in Genetics,* vol. 13, no. 9, pp. 375-376, 1997.

[17] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes," *Nucleic Acids Research,* vol. 28, no. 1, pp. 27-30, Jan. 2000.

[18] K.J. Kauffman, P. Prakash, and J.S. Edwards, "Advances in Flux Balance Analysis," *Current Opinion in Biotechnology,* vol. 14, no. 5, pp. 491-496, 2003.

[19] S. Klamt and E.D. Gilles, "Minimal Cut Sets in Biochemical Reaction Networks," *Bioinformatics,* vol. 20, no. 2, pp. 226-234, 2004.
[20] S. Loewe, "The Problem of Synergism and Antagonism of Combined Drugs," *Arzneimittelforschung,* vol. 3, pp. 285-290, 1953.
[21] P. Pacher, A. Nivorozhkin, and C. Szabo, "Therapeutic Effects of Xanthine Oxidase Inhibitors: Renaissance Half a Century after the Discovery of Allopurinol," *Pharmacological Rev.,* vol. 58, no. 1, pp. 87-114, 2006.
[22] B.O. Palsson, *Systems Biology: Properties of Reconstructed Networks.* Cambridge Univ. Press, 2006.
[23] K.R. Patil, I. Rocha, J. Forster, and J. Nielsen, "Evolutionary Programming as a Platform for in Silico Metabolic Engineering," *BMC Bioinformatics,* vol. 6, no. 308, 2005.
[24] M. Peschel and W. Mende, *The Predator-Prey Model: Do We Live in a Volterra World?* Akademie-Verlag, 1986.
[25] 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,* vol. 103, pp. 249-253, 2002.
[26] 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," *J. Theoretical Biology,* vol. 225, pp. 185-194, 2008.
[27] R. Ramakrishna, J.S. Edwards, A. McCulloch, and B.O. Palsson, "Fluxbalance Analysis of Mitochondrial Energy Metabolism: Consequences of Systemic Stoichiometric Constraints," *Am. J. Physiology-Regulatory, Integrative and Comparative Physiology,* vol. 280, pp. R695-R704, 2007.
[28] D. Rodriguez-Amaya, "Food Carotenoids: Analysis, Composition and Alterations during Storage and Processing of Foods," *Forum of Nutrition,* vol. 56, pp. 35-37, 2003.
[29] M.A. Savageau and E.O. Voit, "Recasting Nonlinear Differential Equations as S-Systems: A Canonical Nonlinear Form," *Math. Biosciences,* vol. 87, pp. 83-115, 1987.
[30] 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.* McGraw-Hill, 2007.
[31] B. Song, P. Sridhar, T. Kahveci, and S. Ranka, "Double Iterative Optimization for Metabolic Network-Based Drug Target Identification," *Int'l J. Data Mining and Bioinformatics,* vol. 3, pp. 124-144, 2007.
[32] P. Sridhar, T. Kahveci, and S. Ranka, "An Iterative Algorithm for Metabolic Network-Based Drug Target Identification," *Proc. Pacific Symp. Biocomputing,* 2007.
[33] P. Sridhar, B. Song, T. Kahveci, and S. Ranka, "OPMET: A Metabolic Network-Based Algorithm for Optimal Drug Target Identification," *Proc. Pacific Symp. Biocomputing,* 2008.
[34] 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'l Microbiology,* vol. 6, no. 2, pp. 127-129, 2003.
[35] A.I. Vogel, A.R. Tatchell, B.S. Furnis, A.J. Hannaford, and P.W.G. Smith, *Vogel's Textbook of Practical Organic Chemistry,* fifth ed., Prentice Hall, 1996.
[36] E.O. Voit, *Computational Analysis of Biochemical Systems: A Practical Guide for Biochemists and Molecular Biologists.* Cambridge Univ. Press, 2000.
[37] E.O. Voit, "Metabolic Modeling: A Tool of Drug Discovery in the Post-Genomic Era," *Drug Discovery Today,* vol. 7, no. 11, pp. 621-628, May 2002.
[38] Y.T. Yang, G.N. Bennett, and K.Y. San, "Genetic and Metabolic Engineering," *Electronic J. Biotechnology,* vol. 1, no. 3, pp. 134-141, 1998.

Bin Song received the BS and MS degrees in computer science from the University of Science and Technology of China, Hefei, in 2002 and 2005, respectively. She is working toward the PhD degree in computer science from the University of Florida from 2006. Her current research interests include bioinformatics and data mining.

İ. Esra Büyüktahtakin received the BS degree in industrial engineering from Fatih University in 2002, the MS degree in industrial engineering from Bilkent University in Turkey 2005, the second MS degree in management science from Lehigh University in 2007, and the PhD degree from the Department of Industrial and Systems Engineering at the University of Florida in August 2009. Following graduation from the University of Florida, she joined the Systems and Industrial Engineering Department at the University of Arizona as a visiting assistant professor. Her research interests include mathematical programming, in particular, mixed integer programming with applications arising in production planning and logistics, supply chain management, bioinformatics, and health care.

Sanjay Ranka received the BTech degree in computer science from the IIT, Kanpur, India, in 1985 and the PhD degree in computer science from the University of Minnesota in 1988. He is currently a professor in the Department of Computer Science. His current research interests are data mining, informatics and grid computing for data intensive applications in high energy physics, bioterrorism and biomedical computing. Most recently, he was the chief technology officer at Paramark where he developed real-time optimization software for optimizing marketing campaigns. He has also held positions as a tenured faculty positions at Syracuse University and as a researcher/visitor at IBM T.J. Watson Research Labs and Hitachi America, Ltd.

Tamer Kahveci received the BS degree in computer engineering and information science from Bilkent University, Turkey, in 1997 and the PhD degree in computer science from the University of California at Santa Barbara in 2004. He is currently an assistant professor in the computer and information science and engineering at the University of Florida. He received the Ralph E. Powe Junior Faculty Enhancement award in 2006, the CSB best paper award in 2008, and the US National Science Foundation (NSF) Career award in 2009. His main research interests are bioinformatics and databases. He has worked on indexing sequence and protein structure databases, sequence alignment and computational analysis of biological pathways.