AN ITERATIVE ALGORITHM FOR METABOLIC NETWORK-BASED DRUG TARGET IDENTIFICATION *

PADMAVATI SRIDHAR, TAMER KAHVECI AND SANJAY RANKA

Department of Computer and Information Science and Engineering,
University of Florida, Gainesville, FL, USA, 32611
E-mail: {psridhar, tamer, ranka}@cise.ufl.edu

Post-genomic advances in bioinformatics have refined drug-design strategies, by focusing on the reduction of serious side-effects through the identification of enzymatic targets. We consider the problem of identifying the enzymes (i.e., drug targets), whose inhibition will stop the production of a given target set of compounds, while eliminating minimal number of non-target compounds. An exhaustive evaluation of all possible enzyme combinations to find the optimal solution subset may become computationally infeasible for very large metabolic networks. We propose a scalable iterative algorithm which computes a sub-optimal solution within reasonable time-bounds. Our algorithm is based on the intuition that we can arrive at a solution close to the optimal one by tracing backward from the target compounds. It evaluates immediate precursors of the target compounds and iteratively moves backwards to identify the enzymes whose inhibition will stop the production of the target compounds while incurring minimum side-effects. We show that our algorithm converges to a sub-optimal solution within a finite number of such iterations. Our experiments on the E.Coli metabolic network show that the average accuracy of our method deviates from that of the exhaustive search only by 0.02 % . Our iterative algorithm is highly scalable. It can solve the problem for the entire metabolic network of Escherichia Coli in less than 10 seconds.

1. Introduction

Traditional drug development approaches focused more on the efficacy of drugs than their toxicity (untoward side effects). Lack of predictive models that account for the complexity of the inter-relationships between the metabolic processes often leads to drug development failures. Toxicity and/or lack of efficacy can result if metabolic network components other than the intended target are affected. This is well-illustrated by the example of the recent failure of *Tolcapone* (a new drug developed for Parkinson's disease) due to observed hepatic toxicity in some patients [9]. Post-genomic drug research focuses more on the identification of specific biological targets (gene products, such as enzymes or proteins) for drugs, which can be manipulated to produce the desired effect (of curing a disease) with minimum disruptive side-effects [20,24]. The main phases in such a drug development strategy are target identification, validation and lead inhibitor identification [7].

*Work supported partially by ORAU (Award no: 00060845). The work of Sanjay Ranka is supported in part by the National Science Foundation under Grant ITR 0325459. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Enzymes catalyze reactions, which produce metabolites (compounds) in the metabolic networks of organisms. Enzyme malfunctions that result in the accumulation of certain compounds may result in diseases. We term such compounds as *Target Compounds* and the remaining compounds as *Non-Target compounds*. For instance, the malfunction of enzyme *phenylalanine hydroxylase* causes buildup of the amino acid, phenylalanine, resulting in phenylketonuria [23], a disease that causes mental retardation. Hence, it is intuitive to identify the optimal enzyme set that can be manipulated by drugs to prevent the excess production of target compounds, with minimal side-effects. We term the side-effects of inhibiting a certain enzyme combination as the *damage* caused to the metabolic network. Formally, we define *damage* of inhibiting an enzyme as the number of non-target compounds whose production is stopped by the inhibition of that specific enzyme.

In our earlier work [22], we developed a graph model for metabolic networks based on the boolean network model [21]. In our model, R, C, and E denote the set of reactions, compounds, and enzymes respectively. The node set consists of all the members of $R \cup C \cup E$. A node is labeled as reaction, compound, or enzyme based on the entity it refers to. Edges represent the interactions in the network. A directed edge from vertex x to vertex y is drawn if one of the following three conditions holds: (1) x represents an enzyme that catalyzes the reaction represented by y. (2) x corresponds to a reactant for the reaction represented by y. (3) x represents a reaction that produces the compound mapped to y. We assume that the inputs to all reactions and compounds are already present in the network and that there are no external inputs.

Figure 1(a) illustrates a small hypothetical metabolic network. A directed edge from an enzyme to a reaction implies that the enzyme catalyzes that reaction. For instance, E_1 catalyzes R_1 and R_2. A directed edge from a compound to a reaction implies that the compound is a reactant (input compound). A directed edge from a reaction to a compound implies that the compound is a product (output compound). In this figure, C_1 is the target compound (i.e., the production of C_1 should be stopped). In order to stop its production, we have to prevent R_1 from taking place. This can be accomplished in two ways: (1) By disrupting one of its catalyzing enzymes (E_1 in this case). Figure 1(b) shows the effects of disrupting E_1. The resulting damage is calculated as the number of non-target compounds whose production is stopped. Since the production of C_2, C_3 and C_4 is stopped, the damage due to the disruption of E_1 is 3. (2) By stopping the production of one of its reactant compounds (C_5 in this case). To stop the production of C_5, we need to recursively look for the enzyme combination which is indirectly responsible for its production (E_2 and E_3). The combined damage of E_2 and E_3 is 1. Thus, the production of the target compound can be stopped by manipulating either E_1 or a combination of E_2 and E_3. The optimal solution is the enzyme combination whose disruption has the minimum damage on the network (E_2 and E_3 in this case).

Problem: Given a large metabolic network and a set of target compounds, we con-

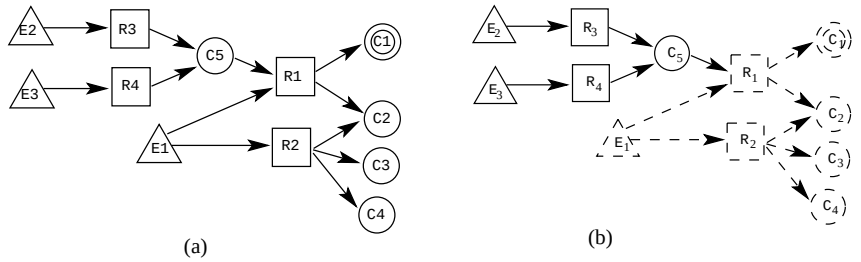


Figure 1. (a)A graph constructed for a metabolic network with four reactions $R_1, \cdots, R_4$, three enzymes E_1, E_2 and E_3, and five compounds $C_1, \cdots, C_5$. Circles, rectangles, and triangles denote compounds, reactions, and enzymes respectively. Here, C_1 (shown by double circle) is the target compound. (b)Effect of inhibiting E_1. Dotted lines indicate the subgraph removed due to inhibition of enzyme E_1.

sider the problem of identifying the set of enzymes whose inhibition eliminates all the target compounds and inflicts minimum damage on the rest of the network. Evaluating all enzyme combinations is not feasible for metabolic networks with a large number of enzymes. This is because, the number of enzyme combinations, i.e., $2^{|E|} - 1$, increases exponentially with the number of enzymes. Efficient and precise heuristics are needed to tackle this problem.

Note that different enzymes and compounds may have varying levels of importance in the metabolic network. Our model simplistically considers all the enzymes and compounds to be of equal importance. It can be extended by assigning weights to enzymes and compounds based on their roles in the network. However, we do not discuss these extensions in this paper.

Contribution: In this paper, we develop a scalable iterative algorithm as an approximation to the optimal enzyme combination detection problem. Our algorithm is based on the intuition that we can arrive at a solution close to the optimal one by tracing backward from the target compounds. It starts by finding the damage incurred due to the removal of each reaction or compound by evaluating its immediate precursors. It then iteratively improves the damage by considering the damage computed for the immediate precursors. It converges when the damage values cannot be improved any further. We prove that the number of iterations is at most the number of reactions on the longest path from any enzyme to the target compounds in the underlying pathway. To the best of our knowledge, this is the first polynomial time solution for a metabolic-network based drug target identification problem.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents the proposed iterative algorithm for determining the enzyme combination whose inhibition achieves the desired effect of inhibiting the production of target compounds. Section 4 presents a theoretical analysis of the algorithm. Section 5 discusses experimental results. Section 6 concludes the paper.

2. Related work

Traditional pharmacological drug discovery approaches involve the incorporation of a large number of hypothetical targets into cell-based assays and automated high throughput screening (HTS) of vast chemical compound libraries[7]. Post-genomic advances in bioinformatics have fostered the development of rational drug-design strategies, that seek to reduce serious side-effects[8,4,3]. This era has brought about the concept of *reverse pharmacology*, in which, the first step is the identification of protein targets, that may be critical intervention points in a disease process[24,20,1]. Since this method is driven by the mechanics of the disease, it is expected to be more efficient than the classical approach[24].

Rapid identification of enzyme (or protein) targets needs a thorough understanding of the underlying metabolic network of the organism affected by a disease. The availability of fully sequenced genomes has enabled researchers to integrate the available genomic information to reconstruct and study metabolic networks[17]. These studies have revealed important properties of metabolic networks[10,2,15]. The potential of an enzyme to be an effective drug target is considered to be related to its essentiality in the corresponding metabolic network[13]. Lemke et. al proposed the measure *enzyme damage* as an indicator of enzyme essentiality [14,16]. Recently, a computational approach to prioritize potential drug targets for antimalarial drugs was developed [18]. A choke-point analysis of *P.falciparcum* was performed to identify essential enzymes which are potential drug targets. The possibility of using enzyme inhibitors as antiparasitic drugs is being investigated through stoichiometric analysis of the metabolic networks of parasites [5,6]. These studies show the effectiveness of computational techniques in reverse pharmacology.

A combination of gene-knockout and micro-array time-course data was used to study the effects of a chemical compound on a gene network [12]. An investigation of metabolite essentiality was carried out with the help of stoichiometric analysis [11]. These approaches underline the importance of studying the role of compounds (metabolites) during the pursuit of computational solutions to pharmacological problems.

3. Iterative algorithm

In this section, we develop a scalable iterative algorithm that finds a sub-optimal solution to the enzyme-target identification problem quickly. Our algorithm is based on the intuition that we can arrive at a solution close to the optimal one, by tracing backwards from the target compounds. We evaluate the immediate precursors of the target compounds and iteratively move backwards to identify the enzymes, whose inhibition will stop the production of the target compounds while incurring minimum damage. Our algorithm consists of an initialization step followed by iterations, until some convergence criteria is satisfied. Let E, R and C denote the sets of enzymes, reactions and compounds of the metabolic network respectively. Let $|E|$, $|R|$ and $|C|$ denote the number of enzymes, reactions and compounds respectively.

The primary data structures are three vectors, namely an *enzyme vector* $V_E = [e_1, e_2, \cdots, e_{|E|}]$, a *reaction vector* $V_R = [r_1, r_2, \cdots, r_{|R|}]$, and a *compound vector* $V_C = [c_1, c_2, \cdots, c_{|C|}]$. Each value, e_i, in V_E denotes the damage of inhibition of enzyme, $E_i \in E$. Each value, r_i, in V_R denotes the damage incurred by stopping the reaction $R_i \in R$. Each value, c_i, in V_C denotes the damage incurred by stopping the production of the compound $C_i \in C$.

Initialization: Here, we describe the initialization of vectors V_E, V_R, and V_C. We initialize V_E first, V_R second, and V_C last.

Enzyme vector: The damage $e_i, \forall i, 1 \leq i \leq |E|$, is computed as the number of non-target compounds whose productions stop after inhibiting E_i. We find the number of such compounds by doing a breadth-first traversal of the metabolic network starting from E_i. We calculate the damage e_i associated with every enzyme $E_i \in E, \forall i, 1 \leq i \leq |E|$, and store it at position i in the enzyme vector V_E.

Reaction vector: The damage r_j is computed as the minimum of the damages of the enzymes that catalyze R_j, $\forall j, 1 \leq j \leq |R|$. In other words, let E_{π_1}, E_{π_2}, $\cdots$, E_{π_k} be the enzymes that catalyze R_j. We compute the damage of r_j as $r_j = \min_{i=1}^{k}\{e_{\pi_i}\}$. This computation is intuitive since a reaction can be disrupted by inhibiting any of its catalyzers. We calculate r_j associated with every reaction $R_j \in R, \forall j, 1 \leq j \leq |R|$ and store it at position j in the reaction vector V_R. Let $E(R_j)$ denote the set of enzymes that produced the damage r_j. Along with r_j, we also store $E(R_j)$. Note that in our model, we do not consider back-up enzyme activities for simplicity.

Compound vector: The damage $c_k, \forall k, 1 \leq k \leq |C|$, is computed by considering the reactions that produce C_k. Let $R_{\pi_1}, R_{\pi_2}, \cdots, R_{\pi_j}$ be the reactions that produce C_k. We first compute a set of enzymes $E(C_k)$ for C_k as $E(C_k) = E(R_{\pi_1}) \cup E(R_{\pi_2}) \cup \cdots \cup E(R_{\pi_j})$. We then compute the damage value c_k as the number of non-target compounds that is deleted after the inhibition of all the enzymes in $E(C_k)$. This computation is based on the observation that a compound disappears from the system only if all the reactions that produce it stop. We calculate c_k associated with every compound $C_k \in C$, $1 \leq k \leq |C|$ and store it at position k in the compound vector V_C. Along with c_k, we also store $E(C_k)$.

Column I_0 in Table 1 shows the initialization of the vectors for the network in Figure 1. The damage e_1 of E_1 is three, as inhibiting E_1 stops the production of three non-target compounds C_2, C_3 and C_4. Since the disruption of E_2 or E_3 alone does not stop the production of any non-target compound, their damage values are zero. Hence, V_E = [3, 0, 0]. The damage values for reactions are computed as the minimum of their catalyzers ($r_1 = r_2 = e_1$ and $r_3 = r_4 = e_2$). Hence, V_R =[3, 3, 0, 0]. The damage values for compounds are computed from the reactions that produce them. For instance, R_1 and R_2 produce C_2. $E(R_1) = E(R_2) = \{E_1\}$. Therefore, $c_2 = e_1$. Similarly c_5 is equal to the damage of inhibiting the set $E(R_3) \cup E(R_4) = \{E_2, E_3\}$. Thus, $c_5 = 1$.

Iterative steps: We iteratively refine the damage values in vectors V_R and V_C in a

Table 1. Iterative Steps: I_0 is the initialization step; I_1 and I_2 are the iterations. V_R and V_C represent the damage values of reactions and compounds respectively computed at each iteration. V_E = [3, 0, 0] in all iterations.

	I_0	I_1	I_2
V_R, V_C	[3, 3, 0, 0], [3, 3, 3, 3, 1]	[1, 3, 0, 0], [1, 3, 3, 3, 1]	[1, 3, 0, 0], [1, 3, 3, 3, 1]

number of steps. At each iteration, the values are updated by considering the damage of the precursor of the precursors. Thus, at nth iteration, the precursors from which a reaction or a compound is reachable on a path of length up to n are considered. We define the length of a path on the graph constructed for a metabolic network as the number of reactions on that path (see Definition 4.2). There is no need to update V_E since the enzymes are not affected by the reactions or the compounds. Next, we describe the actions taken to update V_R and V_C at each iteration. We later discuss the stopping criteria for the iterations.

Reaction vector: Let $C_{\pi_1}, C_{\pi_2}, \cdots, C_{\pi_t}$ be the compounds that are input to R_j. We update the damage of r_j as $r_j = \min\{r_j, \min_{i=1}^{t}\{c_{\pi_i}\}\}$.

The first term of the *min* function denotes the damage value calculated for R_j during the previous iteration. The second term provides the damage of the input compound with the minimum damage found in the previous iteration.

This computation is intuitive since a reaction can be disrupted by stopping the production of any of its input compounds. The damage of all the input compounds are already computed in the previous iteration (say $(n-1)$th iteration). Therefore, at iteration n, the second term of the *min* function considers the impact of the reactions and compounds that are away from R_j by n edges in the graph for the metabolic network. Let $E(R_j)$ denote the set that contains the enzymes that produced the new damage r_j. Along with r_j, we also store $E(R_j)$. We update all $r_j \in V_R$ using the same strategy. Note that the values r_j can be updated in any order, i.e., the result does not depend on the order in which they are updated.

Compound vector: The damage $c_k, \forall k, 1 \leq k \leq |C|$, is updated by considering the damage computed for C_k in the previous iteration and the damages of the reactions that produce C_k. Let $R_{\pi_1}, R_{\pi_2}, \cdots, R_{\pi_j}$ be the reactions that produce C_k. We first compute a set of enzymes as $E(R_{\pi_1}) \cup E(R_{\pi_2}) \cup \cdots \cup E(R_{\pi_j})$. Here, $E(R_{\pi_t})$, $1 \leq t \leq j$, is the set of enzymes computed for R_t after the reaction vector is updated in the current iteration. We then update the damage value c_k as $c_k = \min\{c_k, \text{damage}(\bigcup_{i=1}^{j} E(R\pi_i))\}$.

The first term here denotes the damage value computed for C_k in the previous iteration. The second term shows the damage computed for all the precursor reactions in the current step. Along with c_k, we also store $E(C_k)$, the set of enzymes which provides the current minimum damage c_k.

Condition for convergence: At each iteration, each value in V_R and V_C either remains the same or decreases by an integer amount. This is because a *min* function

is applied to update each value as the minimum of the current value and a function of its precursors. Therefore, the values of V_R and V_C do not increase. Furthermore, a damage value is always an integer since it denotes the number of deleted non-target compounds. We stop our iterative refinement steps when the vectors V_R and V_C do not change in two consecutive iterations. This is justified, because, if these two vectors remain the same after an iteration, it implies that the damage values in V_R and V_C cannot be minimized any more using our refinement strategy.

Columns I_1 and I_2 in Table 1 show the iterative steps to update the values of the vectors V_R and V_C. In I_0, we compute the damage r_1 for R_1 as the minimum of its current damage (three) and the damage of its precursor compound, $c_5 = 1$. Hence, r_1 is updated to 1 and its associated enzyme set is changed to $\{E_2, E_3\}$. The other values in V_R remain the same. When we compute the values for V_C, c_1 is updated to 1, as its new associated enzyme set is $\{E_2, E_3\}$ and the damage of inhibiting both E_2 and E_3 together is 1. Hence, $V_R = [1, 3, 0, 0]$ and $V_C = [1, 3, 3, 3, 1]$. In I_2, we find that the values in V_R and V_C do not change anymore. Hence, we stop our iterative refinement and report the enzyme combination E_2, E_3 as the iterative solution for stopping the production of the target compound, C_1.

Complexity analysis:

Space Complexity: The number of elements in the reaction and compound vectors is $(|R| + |C|)$. For each element, we store an associated set of enzymes. Hence, the space complexity is $O((|R| + |C|) * |E|)$.

Time Complexity: The number of iterations of the algorithm is $O(|R|)$ (see Section 4). The computational time per iteration is $O(G * (|R| + |C|))$, where G is the size of the graph. Hence, the time complexity is $O(|R|G * (|R| + |C|))$.

4. Maximum number of iterations

In this section, we present a theoretical analysis of our proposed algorithm. We show that the number of iterations for the method to converge is finite. This is because the number of iterations is dependent on the length of the longest non-self-intersecting path (see Definitions below) from any enzyme to a reaction or compound.

Definition 4.1. In a given metabolic network, a *non-self-intersecting path* is a path which traces any vertex on the path exactly once. ■

For simplicity, we will use the term *path* instead of *non-self-intersecting path* in the rest of this section.

Definition 4.2. In a given metabolic network, the *length of a path* from an enzyme E_i to a reaction R_j or compound C_k is defined as the number of unique reactions on that path. ■

Note that the reaction R_j is counted as one of the unique reactions on the path from enzyme E_i to R_j.

Definition 4.3. In a given metabolic network, the *preceding path* of a reaction R_j (or a compound C_k) is defined as the length of the longest path from any enzyme in that network to R_j (or C_k). ■

Theorem 4.1. *Let $V_E = [e_1, e_2, \cdots, e_{|E|}]$, $V_R = [r_1, r_2, \cdots, r_{|R|}]$, and $V_C = [c_1, c_2, \cdots, c_{|C|}]$ be the enzyme, reaction and compound vectors respectively (see Section 3). Let n be the length of the longest path (see Definitions 4.2 and 4.1) from any enzyme E_i to a reaction R_j (or a compound C_k). The value r_j (or c_k) remains constant after at most n iterations.* ■

Proof: We prove this theorem by an induction on the number of reactions on the longest path (see Definitions 4.2 and 4.1) from any enzyme in E_i corresponding to $e_i \in V_E$ to C_k.

Basis: The basis is the case when the longest path from an enzyme E_i is of length 1 (i.e., the path consists of exactly one reaction). Let R_j be such a reaction. This implies that there is no other reaction on a path from any E_i to R_j. As a result, the value r_j remains constant after initialization. Let C_k be a compound such that there is at most one reaction from any enzyme to C_k. Let $R_{\pi_1}, R_{\pi_2}, \cdots, R_{\pi_j}$ be the reactions that produce C_k. Because of our assumption there is no precursor reaction to any of these reactions. Otherwise, the length of the longest path would be greater than one. Therefore, the values $r_{\pi_1}, r_{\pi_2}, \cdots, r_{\pi_j}$ and the sets $E(R_{\pi_1}), E(R_{\pi_2}), \cdots, E(R_{\pi_j})$ do not change after initialization. The value c_k is computed as the damage of $E(C_k) = E(R_{\pi_1}) \cup E(R_{\pi_2}) \cup \cdots \cup E(R_{\pi_j})$. Thus, c_k remains unchanged after initialization and the algorithm terminates after the first iteration.

Inductive step: Assume that the theorem is true for reactions and compounds that have a preceding path with at most $n-1$ reactions. Now, we will prove the theorem for reactions and compounds that have a preceding path with n reactions. Assume that R_j and C_k denote such a reaction and a compound. We will prove the theorem for each one separately.

Proof for R_j: Let $C_{\pi_1}, C_{\pi_2}, \cdots, C_{\pi_t}$ be the compounds that are input to R_j. The preceding path length of each of these input compounds, say C_{π_s} is at most n. Otherwise, the preceding path length of R_j would be greater than n.

Case 1: If the preceding path length of C_{π_s} is less than n, by our induction hypothesis, c_{π_s} would remain constant after $(n-1)$th iteration. Thus, the input compound C_{π_s} will not change the value of r_j after nth iteration.

Case 2: If the preceding path length of C_{π_s} is n, then R_j is one of the reactions on this path. In other words, C_{π_s} and R_j are on a cycle of length n. Otherwise, the preceding path length of R_j would be greater than n. Recall that at each iteration, the algorithm considers a new reaction or a compound on the preceding path starting from the closest one. Thus, at nth iteration of computation of r_j, the algorithm completes the cycle and considers R_j. This however will not modify r_j. This is because the value of r_j monotonically decreases (or remains the same) at each iteration. Thus, the initial damage value computed from R_j is guaranteed to be no

better than r_j after $n-1$ iterations. We conclude that r_j will remain unchanged after nth iteration.

Proof for C_k: Let R_{π_1}, R_{π_2}, $\cdots$, R_{π_j} be the reactions that produce C_k. The preceding path length of each of these reactions, say R_{π_s} is at most n. Otherwise, the preceding path length of C_k would be greater than n.

Case 1: If the preceding path length of R_{π_s} is less than n, by our induction hypothesis r_{π_s} would remain constant after $(n-1)$th iteration. Thus, the reaction R_{π_s} will not change the value of c_k after nth iteration.

Case 2: If the preceding path length of R_{π_s} is n, then from our earlier discussion for proof of R_j, r_{π_s} remains unchanged after nth iteration. Therefore R_{π_s} will not change the value of c_k after nth iteration. Hence, by induction, we show that the Theorem 4.1 holds. ■

5. Experimental results

We evaluate our proposed iterative algorithm using the following three criteria:

Execution time: The total time (in milliseconds) taken by the method to finish execution and report if a feasible solution is identified or not.

Number of iterations: The number of iterations performed by the method to arrive at a steady-state solution.

Average damage: The average number of non-target compounds that are eliminated when the enzymes in the result set are inhibited.

We extracted the metabolic network information of Escherichia Coli (E.Coli) from KEGG [19] (`ftp://ftp.genome.jp/pub/kegg/pathways/eco/`). The metabolic network in KEGG has been hierarchically classified into smaller networks according to their functionality. We performed experiments at different levels of hierarchy of the metabolic network and on the entire metabolic network, that is an aggregation of all the functional subnetworks. We devised a uniform labeling scheme for the networks based on the number of enzymes. According to this scheme, a network label begins with 'N' and is followed by the number of enzymes in the network. For instance, 'N20' indicates a network with 20 enzymes. Table 2 shows the metabolic networks chosen, along with their identifiers and the number of compounds (C), reactions (R) and edges (Ed). The edges represent the interactions in the network. For each network, we constructed query sets of sizes one, two and four target compounds, by randomly choosing compounds from that network. Each query set contains 10 queries each.

We implemented the proposed iterative algorithm and an exhaustive search algorithm which determines the optimal enzyme combination to eliminate the given set of target compounds with minimum damage. We implemented the algorithms in Java. We ran our experiments on an Intel Pentium 4 processor with 2.8 GHz clock speed and 1-GB main memory, running Linux operating system.

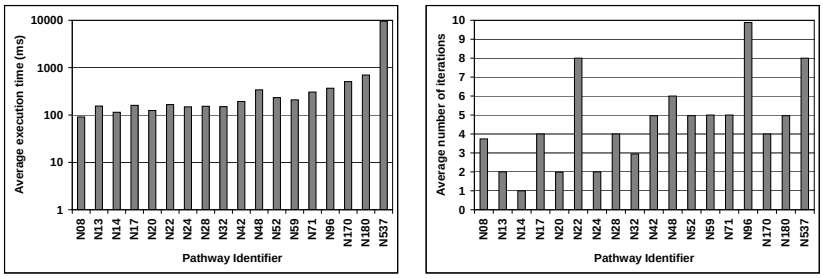
Evaluation of Accuracy: Table 3 shows the comparison of the average damage values of the solutions computed by the iterative algorithm versus the exhaustive

Table 2. Metabolic networks from KEGG with identifier (Id). C, R and Ed denote the number of compounds, reactions and edges (interactions) respectively.

Id	Metabolic Network	C	R	Ed	Id	Metabolic Network	C	R	Ed
N08	Polyketide biosynthesis	11	11	33	N42	Other amino acid	69	63	208
N13	Xenobiotics biodegradation	47	58	187	N48	Lipid	134	196	654
N14	Citrate or TCA cycle	21	35	125	N52	Purine	67	128	404
N17	Galactose	38	50	172	N59	Energy	72	82	268
N20	Pentose phosphate	26	37	129	N71	Nucleotide	102	217	684
N22	Glycan Biosynthesis	54	51	171	N96	Vitamins and Cofactors	145	175	550
N24	Glycerolipid	32	49	160	N170	Amino acid	54	378	1210
N28	Glycine, serine and threonine	36	46	151	N180	Carbohydrate	247	501	1659
N32	Pyruvate	21	51	163	N537	Entire Network	988	1790	5833

Table 3. Comparison of average damage values of solutions determined by the iterative algorithm versus the exhaustive search algorithm.

Pathway Id	$N14$	$N17$	$N20$	$N24$	$N28$	$N32$
Iterative Damage	2.51	8.73	1.63	3.39	1.47	0.59
Exhaustive Damage	2.51	8.73	1.63	3.17	1.47	0.59



(a) (b)

Figure 2. Evaluation of iterative algorithm. (a)Average execution time in milliseconds. (b)Average number of iterations

search algorithm. We have shown the results only upto $N32$, as the exhaustive search algorithm took longer than one day to finish even for $N32$. We can see that the damage values of our method exactly match the damage values of the exhaustive search for all the networks except $N24$. For $N24$, the average damage differs from the exhaustive solution by only 0.02%. This shows that the iterative algorithm is a good approximation of the exhaustive search algorithm which computes an optimal solution. The slight deviation in damage is the tradeoff for achieving the scalability of the iterative algorithm (described next).

Evaluation of Scalability: Figure 2(a) plots the average execution time of our it-

erative method for increasing sizes of metabolic networks. The running time increases slowly with the network size. As the number of enzymes increases from 8 to 537, the running time increases from roughly 1 to 10 seconds. The largest network, $N537$, consists of 537 enzymes, and hence, an exhaustive evaluation inspects $2^{537}-1$ combinations (which is computationally infeasible). Thus, our results show that the iterative method scales well for networks of increasing sizes. This property makes our method an important tool for identifying the right enzyme combination for eliminating target compounds, especially for those networks for which an exhaustive search is not feasible.

Figure 2(b) shows a plot of the average number of iterations for increasing sizes of metabolic networks. The iterative method reaches a steady state within 10 iterations in all cases. The various parameters (see Table 2) that influence the number of iterations are the number of enzymes, compounds, reactions and especially the number of interactions in the network (represented by edges in the network graph). Larger number of interactions increase the number of iterations considerably, as can be seen for networks $N22$, $N48$, $N96$, $N537$, where the number of iterations is greater than 5. This shows that, in addition to the number of enzymes, the number of compounds and reactions in the network and their interactions also play a significant role in determining the number of iterations. Our results show that the iterative algorithm can reliably reach a steady state and terminate, for networks as large as the entire metabolic network of E.Coli.

6. Conclusion

Efficient computational strategies are needed to identify the enzymes (i.e., drug targets), whose inhibition will achieve the required effect of eliminating a given target set of compounds while incurring minimal side-effects. An exhaustive evaluation of all possible enzyme combinations to find the optimal subset is computationally infeasible for large metabolic networks. We proposed a scalable iterative algorithm which computes a sub-optimal solution to this problem within reasonable time-bounds. Our algorithm is based on the intuition that we can arrive at a solution close to the optimal one by tracing backward from the target compounds. We evaluated the immediate precursors of a target compound and iteratively moved backwards, to identify the enzymes, whose inhibition stopped the production of the target compound while incurring minimum damage. We showed that our method converges within a finite number of such iterations. In our experiments on E.Coli metabolic network, the accuracy of a solution computed by the iterative algorithm deviated from that found by an an exhaustive search only by 0.02 %. Our iterative algorithm is highly scalable. It solved the problem for even the entire metabolic network of E.Coli in less than 10 seconds.

References

1. 'Proteome Mining' can zero in on Drug Targets. Duke University medical news, Aug 2004.
2. M Arita. The metabolic world of Escherichia coli is not small. *PNAS*, 101(6):1543–7,

2004.
3. S. Broder and J. C. Venter. Sequencing the Entire Genomes of Free-Living Organisms: The Foundation of Pharmacology in the New Millennium. *Annual Review of Pharmacology and Toxicology*, 40:97–132, Apr 2000.
4. S. K. Chanda and J. S. Caldwell. Fulfilling the promise: Drug discovery in the post-genomic era. *Drug Discovery Today*, 8(4):168–174, Feb 2003.
5. A. Cornish-Bowden. Why is uncompetitive inhibition so rare? *FEBS Letters*, 203(1):3–6, Jul 1986.
6. A. Cornish-Bowden and J. S. Hofmeyr. The Role of Stoichiometric Analysis in Studies of Metabolism: An Example. *Journal of Theoretical Biology*, 216:179–191, May 2002.
7. J Drews. Drug Discovery: A Historical Perspective. *Science*, 287(5460):1960–1964, Mar 2000.
8. Davidov et. al. Advancing drug discovery through systems biology. *Drug Discovery Today*, 8(4):175–183, Feb 2003.
9. Deane et. al. Catechol-o-methyltransferase inhibitors versus active comparators for levodopa-induced complications in parkinson's disease. *Cochrane Database of Systematic Reviews*, 4, 2004.
10. Hatzimanikatis et. al. Metabolic networks: enzyme function and metabolite structure. *Current Opinion in Structural Biology*, (14):300–306, 2004.
11. Imielinski et. al. Investigating metabolite essentiality through genome scale analysis of E. coli production capabilities. *Bioinformatics*, Jan 2005.
12. Imoto et. al. Computational Strategy for Discovering Druggable Gene Networks from Genome-Wide RNA Expression Profiles. In *PSB 2006 Online Proceedings*, 2006.
13. Jeong et. al. Prediction of Protein Essentiality Based on Genomic Data. *ComPlexUs*, 1:19–28, 2003.
14. Lemke et. al. Essentiality and damage in metabolic networks. *Bioinformatics*, 20(1):115–119, Jan 2004.
15. Ma et. al. Decomposition of metabolic network into functional modules based on the global connectivity structure of reaction graph. *Bioinformatics*, 20(12):1870–6, 2004.
16. Mombach et. al. Bioinformatics analysis of mycoplasma metabolism: Important enzymes, metabolic similarities, and redundancy. *Computers in Biology and Medicine*, 2005.
17. Teichmann et. al. The Evolution and Structural Anatomy of the Small Molecule Metabolic Pathways in Escherichia coli. *JMB*, 311:693–708, 2001.
18. Yeh et. al. Computational Analysis of Plasmodium falciparum Metabolism: Organizing Genomic Information to Facilitate Drug Discovery. *Genome Research*, 14:917–924, 2004.
19. M Kanehisa and S Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, 28(1):27–30, Jan 2000.
20. C Smith. Hitting the target. *Nature*, 422:341–347, Mar 2003.
21. R. Somogyi and C.A. Sniegoski. Modeling the complexity of genetic networks: Understanding multi-gene and pleiotropic regulation. *Complexity*, 1:45–63, 1996.
22. P. Sridhar, T. Kahveci, and S. Ranka. Opmet: A metabolic network-based algorithm for optimal drug target identification. Technical report, CISE Department, University of Florida, Sep 2006.
23. R. Surtees and N. Blau. The neurochemistry of phenylketonuria. *European Journal of Pediatrics*, 159:109–13, 2000.
24. Takenaka T. Classical vs reverse pharmacology in drug discovery. *BJU International*, 88(2):7–10, Sep 2001.