OPMET: A metabolic network-based algorithm for optimal 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

{psridhar, tamer, ranka}@cise.ufl.edu

Abstract

Recent advances in bioinformatics promise rational drug-design methods which will reduce serious side-effects through the identification of enzymatic targets. Efficient computational methods are required to identify the optimal enzyme-combination (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 may become computationally infeasible for very large metabolic networks.

*Work supported partially by ORAU (Award No: 00060845).

We formulate the optimal enzyme-combination identification problem as an optimization problem on metabolic networks. We define a graph based computational model of the network which encapsulates the impact of enzymes onto compounds. We propose a branch-and-bound algorithm, named *OPMET*, to explore the search space. We develop a cost model and two enzyme prioritization strategies, Static OPMET and Dynamic OPMET, based on it. Static OPMET prioritizes enzymes according to their impacts, such that the most promising enzymes are inspected first for possible inclusion in the optimal subset. Dynamic OPMET dynamically updates the priorities as the search space is explored. We also develop two filtering strategies to prune the search space while still guaranteeing an optimal solution. They compute an upper bound to the number of target compounds eliminated and a lower bound to the side-effect respectively. Our experiments on E.Coli metabolic network show that OPMET can reduce the total search time by several orders of magnitude as compared to the exhaustive search.

1 Introduction

In pharmaceutics, the development of every drug mainly involves target identification, validation and lead inhibitor identification [5]. Traditional drug discovery 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 [7]. Post-genomic advances has seen a perspective shift in drug research. The current focus is on the identification of 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 [22, 25].

Enzymes catalyze reactions, which produce metabolites (compounds) in the metabolic networks of organisms. Enzyme malfunctions can result in the accumulation of certain compounds which 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 [24], a disease that causes mental retardation. It is therefore intuitive to locate the optimal enzyme set which 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.

Given a metabolic network and a set of target compounds, we consider the problem of identifying the set of enzymes whose inhibition eliminates the target compounds and incurs minimum damage. Evaluating all enzyme combinations is not feasible as the number of such combinations increases exponentially with the number of enzymes. Hence, more efficient computational methods are needed. We propose *OPMET*, an **Op**timal enzyme drug target identification algo-

rithm based on **Met**abolic networks, to solve this problem. The main contributions of our paper are as follows:

1. We define a graph based computational model of the metabolic network that encapsulates the impact of enzymes onto compounds. We formulate the optimal enzyme combination identification problem as an optimization problem on this graph. We develop a cost model that takes both the observed and potential damage (i.e, side-effects) resulting from the inhibition of an enzyme set into account, for the cost computation.

2. We propose a branch-and-bound algorithm, named *OPMET*, to explore the search space. We develop two enzyme prioritization strategies, *Static OPMET* and *Dynamic OPMET* based on the cost model. Static OPMET prioritizes enzymes according to the impact of their inhibition on the production of compounds in the metabolic network. It inspects the most promising enzymes first, for possible inclusion in the optimal subset. Dynamic OPMET dynamically updates the priorities as the search space is explored.

3. We develop two filtering strategies for OPMET, namely *target filter* and *non-target filter*, to prune the search space while still guaranteeing an optimal solution. The target filter eliminates a subspace when it is proven that there is no combination of enzymes in this space that can stop the production of all the target compounds (i.e., there is no useful drug target). The non-target filter prunes subspaces where there is no solution with a damage less than that of the optimal solution found so far.

Our experiments on E.Coli metabolic network (data extracted from KEGG [18, 19]) show that our methods can reduce the total search time by several orders of magnitude as compared to the exhaustive search. Dynamic OPMET with a combination of the target and non-target filters prunes 91.6 % of the search space of the exhaustive search (2^{32} combinations) on average. Dynamic OPMET with combined filters generates the optimal enzyme combination within the exploration 0.005 % of the search space on average.

The rest of the paper is organized as follows. Section 2 formally defines the problem and describes our proposed network model. Section 3 presents the proposed OPMET algorithm with static and dynamic prioritization and filtering strategies. Section 4, discusses experimental results. Section 5 discusses the related work. Section 6 gives a brief discussion of the contribution of the paper.

2 Problem Definition

In this section, we propose a formal definition of the optimal enzyme combination identification problem and describe our model of the metabolic network.

We develop a graph based model that captures the interactions between reactions, compounds, and enzymes. Our model is a variation of the boolean network model [23, 20]. Let R, C, and E denote the set of reactions, compounds, and enzymes respectively. The vertex set consists of all the members of $R \cup C \cup E$. A vertex is labeled as reaction, compound, or enzyme based on the entity it refers to. Let V_R, V_C, and V_E denote the set of vertices from R, C, and E. A unique identifier is also stored along with each vertex. A directed edge from vertex x to

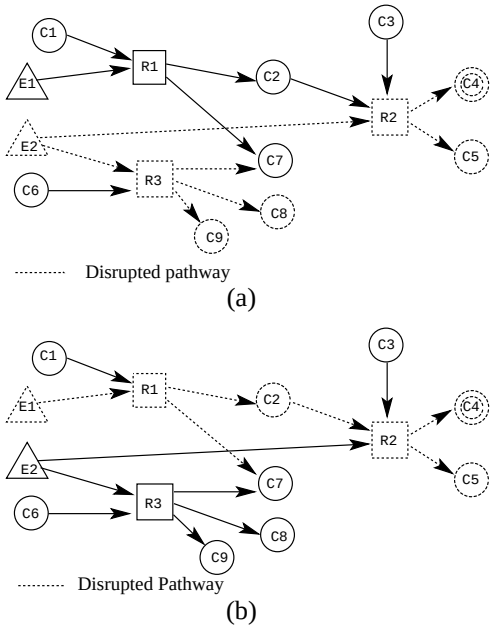


Figure 1: A graph constructed for a metabolic network with three reactions R_1, R_2, and R_3, two enzymes E_1 and E_2, and nine compounds C_1, C_2, $\cdots$, C_9. Circles, rectangles, and triangles denote compounds, reactions, and enzymes respectively. Here, C_4 (shown by double circle) is the target compound. Dotted lines indicate the subgraph removed due to inhibition of an enzyme. (a) Effect of inhibiting E_2. (b) Effect of inhibiting E_1.

vertex y is then 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 substrate for the reaction represented by y. (3) x represents a reaction that produces the compound mapped to y. Notice that an edge is drawn only between a vertex in V_R and a vertex in $V_C \cup V_E$.

Figure 1 illustrates a small hypothetical metabolic network. A directed edge from an enzyme to a reaction implies that the enzyme catalyzes the reaction (i.e., E_1 catalyzes R_1 and E_2 catalyzes R_2 and R_3). A directed edge from a compound to a reaction implies that the compound is a reactant. A directed edge from a reaction to a compound implies that the compound is a product. In this figure, C_4 is the target compound (i.e., the production of C_4 should be stopped). In order to stop the production of C_4, R_2 has to be prevented from taking place. This can be achieved in two ways. One way is by disrupting one of its catalyzing enzymes (E_2 in this case). Another is by stopping the production of one of its reactant compounds (C_2 or C_3 in this case). If we choose to stop the production of C_2, we need to recursively look for the enzyme which is indirectly responsible for its production (E_1 in this case). Thus, the production of the target compound can be stopped by manipulating either E_1 or E_2.

Figure 1(a) shows the disruption of E_2 and its effect on the network. Inhibiting E_2 results in the knock out of compounds C_5, C_8 and C_9 in addition to the target compound, C_4. Note that the production of C_7 is not stopped since it is produced by R_1 even after the inhibition of E_2. We term the effect of disrupting non-target compounds as the ***side-effect*** of manipulating E_2. We define the number of non-

target compounds knocked out as the *damage*, the manipulation of an enzyme set causes to the metabolic network. In this case, the damage of inhibiting E_2 is 3 (i.e., C_5, C_8 and C_9). Figure 1(b) shows the inhibition of E_1 and its effect on the same network. In this case, the damage is 2 (i.e., C_2 and C_5). The important observation is that E_1 and E_2 both achieve the effect of disrupting the target compound, C_4. Hence, E_1 and E_2 are both potential drug targets. However, E_1 is a better drug-target than E_2 since it causes lesser damage.

We formally state the optimal enzyme combination identification problem as:

> Given a set of target compounds T ($T \subset C$), find the set of enzymes X ($X \subseteq E$) with minimum damage, whose inhibition stops the production of all the compounds in T.

Our graph model accurately reflects the impacts of gene manipulations. Inhibition of a set of enzymes, for example, is simulated by deletion of the vertices for the enzymes in that set. This modification then propagates to the vertices that are reachable from the deleted vertices.

Lemke et. al [13, 15] defined the damage of inhibition of an enzyme as the number of compounds whose production stops after the inhibition of that enzyme (all the compounds are assumed to be of equal importance to the network). Our definition of *damage* is similar to that of Lemke in principle. We differ by excluding the target compounds from the damage computation. At a high level, we consider damage as the side-effects of inhibiting an enzyme set.

3 Methods

The number of possible subsets of enzymes that need to be considered for finding the optimal drug target is exponential in an exhaustive search. Clearly, this is not feasible beyond small sized metabolic networks. In this section, we propose OPMET, a branch and bound algorithm that considerably reduces the number of possible combinations to be searched while still guaranteeing to find an optimal solution. We first describe the state space that represents all possible solutions using a tree and the backtracking mechanism which guarantees that the search is conducted till an optimal solution is found. As part of any effective branch and bound strategy it is important to find a good solution quickly. This calls for effective filtering (or pruning) of subspaces that can be guaranteed not to have better solution that the best found so far. Our prioritization and filtering strategies which achieve this goal are described in detail in the following subsections.

3.1 State Space and Basic Strategy

In this subsection we discuss our modeling of the search space and the basic underlying strategy of the OPMET algorithm. We develop a systematic and flexible enumeration of the search space and use it to build our efficient search strategy to find optimal result.

Let $E_i, \forall i, 1 \leq i \leq m$ denote the set of enzymes for a metabolic network. The search space is modeled as a tree structure. Every node of this tree corresponds to a state in the search space and it is represented by a 4-tuple $([e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_m}], k, d, remove)$. Here, $\pi_1, \cdots, \pi_m$ is a permutation of 1, 2, $\cdots$, m. The first parameter

corresponds to the state of all the enzymes (i.e., e_{π_i} corresponds to enzyme E_i). $e_{\pi_i} = 1$ if E_i is inhibited. Otherwise, $e_{\pi_i} = 0$. The parameter k indicates that the first k enzymes are considered at that search state. The decision to inhibit or not inhibit has been fixed for enzymes from 1 to $k-1$. We now set $e_{\pi_k} = 1$ and $e_{\pi_i} = 0$, $\forall i$, $k < i \leq m$. The damage incurred due to inhibited enzymes at that state is represented by d. The final parameter, $remove$, is a boolean variable. It takes value *True* if the inhibited enzymes stop the production of all the target compounds. Otherwise, it is set to *False*. We call a node with $remove =$ True as a *true node*, and a node with $remove =$ False as a *false node*. Figure 2 shows part of the search tree for a hypothetical 4-enzyme network. The tuple for node n_1 indicates that enzyme E_1 is inhibited and it is a true node with damage $d = 5$. $k = 1$ since the decision to inhibit or not has been fixed only for one enzyme.

OPMET Algorithm: We start with a *root node* whose 4-tuple is ($[0, 0, \cdots, 0]$, 0, 0, False) indicating that all enzymes are present in the network. As the search space is traversed, we keep the true node with the minimum damage found so far as the *current true solution* and store the associated damage value as D, the *global cut-off threshold*. D is initialized to the number of compounds in the network. At any point, we have an *active set* of nodes A, stored in a stack structure. A contains the nodes currently being considered. Let node $N = ([e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_m}], k, d, remove)$ be the node on top of this stack (i.e., the node to be evaluated). We check if the damage of the current node N, $d < D$ and whether it is a true node. There are three cases:

- *Case 1:* N is a true node with damage $d < D$. In this case, we save N as the

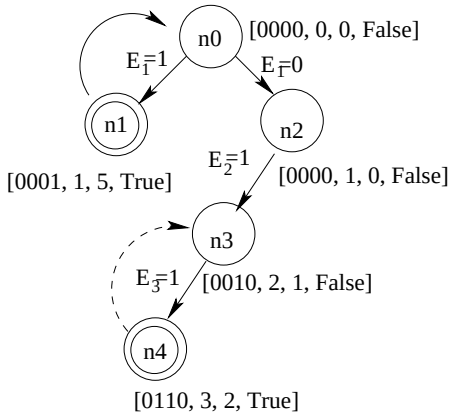


Figure 2: The basic OPMET strategy for a hypothetical 4-enzyme network. Enzymes are ordered as E_1, E_2, E_3, E_4. n_0, n_1, $\cdots$, n_4 are the nodes generated. The initial *global cut-off threshold* $D = 10$ (initialized to the total number of compounds in the network). n_1 is a true solution (shown by double circle) with damage $d = 5$. Since $d < D$, D is updated to 5. n_1 is saved and the subtree rooted at n_1 is pruned. The method backtracks to n_0. n_2, n_3 are false nodes generated along the search with damage $d < D$. n_5 is a true node with $d = 2$. As $d < D$, D is updated to 2 and the method backtracks to search the unexplored space for solutions with $d < D$ (indicated by the dashed edge).

current true solution and update D with the damage value of N. We then backtrack.

- *Case 2:* N is a true or false node with damage $d > D$. In this case we prune the subtree rooted at N. We then backtrack.

- *Case 3:* N is a false node with damage $d < D$. In this case, we insert N in the active set A for backtracking purposes. We then create a new node N' by setting $e_{\pi_{k+1}} = 1$ in N (i.e., we inhibit the enzyme $E_{\pi_{k+1}}$). The resulting node is $N' = ([e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_m}], k + 1, d', remove')$. The node N' is evaluated in the next step similarly.

Backtracking involves following steps. First we pick the top node from the ac-

tive nodes stack A. Let $N' = ([e'_{\pi_1}, e'_{\pi_2}, \cdots, e'_{\pi_m}], k', d', remove')$ denote this node. We then set $e_{\pi_{k'+1}} = 0$ (indicating the node we are backtracking from) and $e_{\pi_{k'+2}} = 1$ in N' (i.e., we inhibit the enzyme $e_{\pi_{k'+2}}$). The resulting node becomes the node to be evaluated in the next step. The first two cases above stop expanding the tree at the current node. The former one implies that the current node is a possible solution (node n_1 in Figure 2). The latter one implies that the current node incurs too much damage to lead to a possible solution. The third case happens when the current node does not stop production of all the target compounds, but the damage is lower than the damage of the current best solution (nodes n_2 and n_3 in Figure 2). Such nodes may produce a possible solution (node n_4 in Figure 2) with the inhibition of more enzymes. Thus, they need to be explored further to ensure that we find an optimal solution. The search terminates when there are no more nodes to explore. At this stage, the current true solution is the *optimal solution*.

3.2 OPMET Prioritization Strategies

In this subsection we present our cost model as the basis for enzyme evaluation and our resulting enzyme prioritization strategies. The purpose of ordering enzymes by a priority measure is to test the combinations with a high likelihood of deleting the target compounds first. We develop two enzyme prioritization strategies: **1) Static OPMET:** We pre-order the enzymes according to a priority measure and apply the OPMET algorithm. **2) Dynamic OPMET:** We dynamically select the next enzyme to inhibit based on the partial solution. These are

described in more detail in the rest of the section.

3.2.1 Static OPMET

The simplest way to order the enzymes is to sort them in ascending order of damage. However, this is inaccurate for several reasons. First, since damage is defined in terms of non target compounds, it does not reflect an enzyme's impact on the target compounds. Second, the inhibition of an enzyme can make a compound more vulnerable even if the compound is not removed entirely. For example, in Figure 1, the production of C_7 is not stopped after the inhibition of E_1 or E_2 individually. However, C_7 is removed if E_1 is inhibited as well as E_2. Thus, damage values of individual enzymes are not good indicators of the combined damage of these enzymes.

Cost Model: We develop a cost model as the basis for enzyme ordering in OPMET. This cost model takes both the observed and potential damage (side-effects) resulting from the inhibition of an enzyme set into the cost computation. For each enzyme $E_i \in E$, the set of enzymes in the network, we compute a weight $W(E_i)$ as $W(E_i) = 0$ if E_i inhibited and $W(E_i) = 1$ otherwise. We assign rational weights (fractions between 0 and 1) to the reaction and compound nodes and the edges. Intuitively, the weight of a node or edge denotes the rate at which that node or edge appears in the network. We then use these weights to compute the cost of E_i. The weight of each node depends on the weights of the incoming edges and the type of the node (i.e., reaction or compound):

- **Cost Rule 1:** Let R_j be a reaction node. Let w_i, $1 \leq i \leq k$, denote the weights of the incoming edges to R_j. We compute the weight of R_j as:

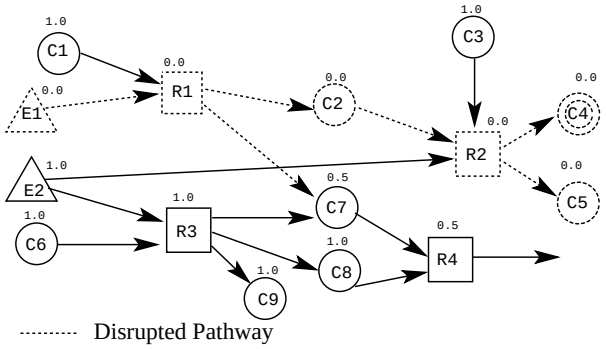


Figure 3: Effect of deletion of enzyme E_1 on the weights of reactions and compounds in the network.

$$W(R_j) = \min_{i=1}^{k}\{w_i\}.$$

This computation is intuitive since a reaction takes place only if all the inputs are present.

- **Cost Rule 2:** Let C_j be a compound node. Let w_i, $1 \leq i \leq k$, denote the weights of the incoming edges to C_j. We compute the weight of C_j as:

  $$W(C_j) = \frac{1}{k}\sum_{i=1}^{k}\{w_i\}.$$

  This computation captures the property that a compound disappears from the system only if all the reactions that produce it stop.

We define the weight of an edge as the weight of the node for which it is the outgoing edge.

In order to compute the cost of E_i, we set the weight of E_i to zero (i.e., $W(E_i) = 0$). The weights of all the reaction and compound nodes are assigned progressively by a breadth-first search, according to the above scheme. The weights of all the nodes and edges which can be reached from E_i are recomputed to reflect the change. The effect of deleting E_1 on the weights of reactions

and compounds in the network is shown in Figure 3. We define an *impact vector* for each enzyme based on the effects of its inhibition.

Definition 1 *Given a network with n compounds, C_j, $1 \leq j \leq n$. Let $W_i(C_j)$ denote the weight of the node corresponding to C_j after the inhibition of enzyme E_i. We define the* impact vector *of E_i as $I(E_i) = [W_i(C_1), W_i(C_2), \cdots, W_i(C_n)]$. We term $W_i(C_j)$ as the* impact of E_i on C_j, $\forall j$. ■

The impact vector of an enzyme approximates the amount of each compound that remains after the inhibition of that enzyme. Every entry of the impact vector is a fractional number between 0 and 1, where 0 indicates that the corresponding compound does not exist after inhibition of the corresponding enzyme. We define the cost of inhibiting an enzyme as follows:

Definition 2 *Given a network with n compounds, C_j, $1 \leq j \leq n$. Assume that the compounds C_j, $\forall j$, $1 \leq j \leq k \leq n$ constitute the set of target compounds. Assume that the remaining compounds C_j, $\forall j$, $k+1 \leq j \leq n$ constitute the non-target compounds. Let $I(E_i) = [W_i(C_1), \cdots, W_i(C_n)]$ denote the impact vector of E_i. We define the cost of inhibiting E_i as*

$$\textit{cost}(E_i) = I(E_i) \cdot V^T,$$

where $V = [v_1, \cdots, v_n]$ is the normalization vector*: $v_i = \frac{n-k}{k}$ for $1 \leq i \leq k$, and $v_i = -(\frac{n-k}{n-k})$ for $k < i \leq n$.* ■

Each target compound contributes a positive value and each non-target com-

pound contributes a negative value to the cost of an enzyme. The magnitude of the contribution of a compound increases linearly with the amount of that compound that diminishes from the system after the inhibition of the corresponding enzyme. In other words, if a target compound is eliminated, the cost is small. On the other hand, if a non-target compound is eliminated, the cost is large. This is justified since the cost promotes removal of target compounds and demotes the removal of non-target compounds.

In order to benefit from the pruning power of OPMET cases 1 and 2 (see Section 3.1), we need to compute the permutation $\pi_1, \cdots, \pi_m$ carefully. The earlier we place the enzymes in the optimal solution in this permutation, the better, as OPMET reaches the optimal solution earlier under such an ordering. Reaching the optimal solution early increases the chances of pruning the remaining nodes of the search tree due to OPMET Case 2. We sort the enzymes in the ascending order of cost (see Definition 2). This is justified since a small cost value indicates that the corresponding enzyme has a large impact on target compounds and low impact on non-target compounds.

3.2.2 Dynamic OPMET

Static OPMET prioritizes enzymes based on their individual potential to be part of the solution. However, this model does not fully account for the combined damage of a set of enzymes. Finding the combined damage of a set of enzymes is non trivial. This is because the combined damage depends on the overlap of the reactions catalyzed by these enzymes as well as their individual damages. Enzymes that catalyze almost the same set of reactions have less combined damage

1. Let $N = ([e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_m}], k, d, remove)$ be the node currently being evaluated.
2. Let $R = [r_1, r_2, \cdots, r_n]$ /* $r_i \in [0, 1]$ corresponds to the remaining fraction of compound C_i, $\forall i$ */
3. For every candidate enzyme $e_i \in \{e_{\pi_k}, e_{\pi_{k+1}}, \cdots, e_{\pi_m}\}$
   (a) $R_i = R \odot I(E_i)$ /* Compute new ratio */
   (b) Cost(E_i) $= R_i \cdot V^T$ /* Cost of inhibiting E_i */
4. Let $j = \arg\min_j \{\text{Cost}(E_j)\}$
5. $R = R_j$ /* Update the ratio after inhibition of the enzyme with least cost */
6. Select E_j as the next enzyme to inhibit.

Figure 4: Dynamic OPMET procedure for enzyme evaluation.

compared to enzymes that catalyze disjoint reaction sets. This is due to the overlap of the damage of the enzymes in the former case. We develop a dynamic strategy, Dynamic OPMET, which sorts the enzymes on the fly, based on the partial solution. At a high level, this strategy evaluates the current state at every step and picks the enzyme-to-inhibit for the next step based on this evaluation.

Our goal is to dynamically decide which enzyme to inhibit next based on the combined impact of all the enzymes inhibited so far. We predict the remaining fraction of all the compounds after inhibition of each enzyme E_i, with the help of its impact vector, $I(E_i)$ (see Definition 1). This procedure is described in Figure 4. Let $R = [r_1, r_2, \cdots, r_n]$ denote the remaining fractions of compounds (step 1). Here, $r_i \in [0, 1]$ corresponds to compound C_i, $\forall i$. We initialize $r_i = 1\ \forall C_i$ indicating that all compounds are being produced without any disruption in the metabolic network. Let V be the normalization vector (see Definition 2).

At every step of the Dynamic OPMET algorithm, let $N = ([e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_m}], k, d, remove)$ be the node currently being evaluated (i.e., the decision to inhibit or not inhibit has been fixed for $e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_{k-1}}$) (step 2). We now need to decide which enzyme has to be evaluated next. In Step 3a, for every enzyme in the remaining enzyme set ($e_{\pi_i}, \forall i, k \leq i \leq m$), we compute the new remaining fractions of compounds (R_i). This is done by a Vector Direct Product of R and the impact vector of E_i ($I(E_i)$). Vector direct product is defined as $X \odot Y = [x_1y_1, x_2y_2, \cdots, x_ny_n]$, where $X = [x_1, \cdots, x_n]$ and $Y = [y_1, \cdots, y_n]$. The resulting vector R_i is an approximation to the impact of inhibition of the enzyme E_i in addition to already inhibited enzymes. This is justified since the quantity of a compound eliminated by a combination including E_i will be at least as much as the quantity eliminated by E_i alone. A good candidate enzyme at this step is one which ensures that lesser of the target compounds remain after its inhibition. Also, it should ensure that the non target compounds suffer the minimum possible damage. Our cost model satisfies these requirements. In Step 3b, we compute the cost of each enzyme E_i as the dot product of R_i and V. In step 4, we pick the enzyme with the minimum cost (E_j). We update R with the amounts of remaining compounds in vector R_j. E_j is chosen as the next enzyme to inhibit. Thus, this strategy dynamically chooses the the next best enzyme to inhibit.

The cost of finding the best enzyme at each step is $O(mn)$, where m and n denote the number of enzymes and compounds in the metabolic network respectively. This is because a vector direct product costs $O(n)$, and $O(m)$ such products

are carried out.

3.3 OPMET Filtering Strategies

The static and dynamic OPMET strategies perform a cost-based ordering of the enzymes in order to arrive at the optimal set of enzymes quickly. However, the method still needs to inspect a large portion of the search space, in order to ensure that the result is the optimal one. A good filtering strategy is essential to ensure optimality within a reasonable amount of time. We propose two strategies to eliminate large portions of the search space quickly. We start by introducing the following theorem (proof given at the end of this subsection) which establishes a relationship between the impact of enzymes and their damage.

Theorem 1 *Let $E = \{E_1, E_2, \cdots, E_r\}$ be a set of enzymes. Let C_j be a compound in the metabolic network. Let $d_i(C_j)$, $1 \leq i \leq r$, denote the impact of E_i on C_j. If the inhibition of all the enzymes in E stops the production of C_j, then $\sum_{i=1}^{r}(1 - d_i(C_j)) \geq 1$.* ■

Next, we describe our filtering strategies.

Target Filter: We traverse the state space to locate the optimal combination of enzymes to delete a given target compound set. Considerable effort is spent evaluating combinations which might yield a solution as the encountered damage value is less than D, the global cut-off threshold (see Subsection 3.1). If none of these combinations ultimately delete the target compound set, the effort spent is wasted. Finding such combinations will improve the performance of the search vastly.

This is the motivation behind our *Target filter*.

The target filter eliminates a bulk of the search space when it is proven that there is no combination of enzymes in this space that can stop the production of all the target compounds (i.e., there is no useful drug target). This filtering strategy is based on Theorem 1. Formally, let node $N = ([e_{\pi_1}, e_{\pi_2}, \cdots, e_{\pi_m}], k, d,$ False) be a node in the search space. Let T denote the set of target compounds. Backtrack if

$$\sum_{i=1}^{k}(1 - d_i(C))e_{\pi_i} + \sum_{i=k+1}^{m}(1 - d_i(C)) < 1, \exists C \in T.$$

In this inequality, the first term indicates the impact of enzymes, which are currently part of the solution set, on the target compounds. The second term represents the impact of the remaining enzymes on the target compounds. Thus, this filter examines not only the next enzyme being considered, but all the enzymes which might eventually be considered as part of the solution.

Non-target Filter: We traverse the state space to locate the optimal combination of enzymes to delete a given target compound set. The motivation behind this filtering strategy is to quickly determine if there is any solution in the subtree with a damage $d < D$, the global cut-off threshold (see Subsection 3.1). This strategy eliminates a large chunk of the search space where all the possible combinations have damage more D. This filter utilizes Theorem 1 similar to the target filter. The idea is as follows. At a given node N, for each target compound, C, we find

the minimum number of enzymes, n such that

$$\sum_{i=1}^{n}(1 - d_i(C))e_{\pi_i} \geq 1.$$

This gives us the minimum number of enzymes needed to delete C. Let n_{max} be the maximum value of n for any target compound (i.e, we will need at least n_{max} enzymes to delete the entire target compound set). Now, we sort the remaining enzymes (enzymes not considered so far) in the ascending order of their damage values. Let d_{max} be the damage of the enzyme at index n_{max}. If d_{max} in addition to the damage incurred so far is greater than D, the global cut-off threshold, we prune the sub tree rooted at N.

Proof of Theorem 1: Let $E = \{E_1, E_2, \cdots, E_r\}$ be a set of enzymes. Let C_j be a compound in the metabolic network. Let d_i, $1 \leq i \leq r$, denote the impact of E_i on C_j (see Definition 1). Let $d_i(R_k)$, $1 \leq i \leq r$, denote the impact of E_i on R_k. We need to show that the following rules hold:

Rule 1: If the inhibition of all the enzymes in E stops the production of C_j, then

$$\sum_{i=1}^{r}(1 - d_i(C_j)) \geq 1.$$

Rule 2: If the inhibition of all the enzymes in E stops a reaction R_k, then

$$\sum_{i=1}^{r}(1 - d_i(R_k)) \geq 1.$$

We first consider the case when a reaction R_k is stopped and prove that Rule 2 holds, given that Rule 1 is correct.

Case 1: $\exists\, E_j \in E$ such that E_j catalyzes R_k. Hence, the inhibition of E_j stops R_k. The impact of E_j on R_k is $1-d_j(R_k) = 1$. Therefore, $\sum_{i=1}^{r}(1-d_i(R_k)) \geq 1$. *Rule 2 holds*.

Case 2: No enzyme $E_j \in E$ catalyzes R_k. This implies that E removes C_i, which is an input to R_k. From Rule 1, $\sum_{i=1}^{r}(1 - d_i(C_i)) \geq 1$. From the Cost Model (see Cost Rule 1 in subsection 3.2), $d_i(R_k) = \min\{d_i(C')\}$, C' is the set of all input compounds of R_k. Thus, $1 - d_i(R_k) = \max\{1 - d_i(C')\}$. Hence,

$\sum_{i=1}^{r}(1 - d_i(R_k)) \geq \sum_{i=1}^{r}(1 - d_i(C_i)) \geq 1$.

Rule 2 holds.

We now consider the case when the production of a compound C_j stops and prove that Rule 1 holds, given that Rule 2 is correct. If E removes C_j, it means that all the reactions that produce C_j have been stopped. Let $R_1, R_2, \cdots, R_t$ be the reactions that produce C_j. For each reaction $R_k, 1 \leq k \leq t$, one of the the following two cases has to be true.

Case 1: $\exists\, E_j \in E$ such that E_j catalyzes R_k. Hence, the inhibition of E_j stops R_k. The impact of E_j on R_k is $1 - d_j(R_k) = 1$. From the Cost Model (see Cost Rule 2 in subsection 3.2), $1 - d_j(C_j) \geq \frac{1}{t}$. Since all the t input reactions of C_j have been stopped,

$\sum_{i=1}^{r}(1 - d_i(C_j)) \geq \frac{1}{t} \times t \geq 1.$

Rule 1 holds.

Case 2: No enzyme $E_j \in E$ directly catalyzes R_k. But since R_k is stopped by the inhibition of enzymes in E, $\sum_{i=1}^{r}(1 - d_i(R_k)) \geq 1$ (Rule 2). Also given that t reactions produce C_j,

$$(1 - d_i(C_j)) = 1 - \frac{\sum_{j=1}^{t}(1-d_i(R_j))}{t}$$

$$\sum_{i=1}^{r}(1 - d_i(C_j)) = r - \frac{1}{t}\sum_{i=1}^{r}\sum_{j=1}^{t}(1 - d_i(R_j))$$

$$\geq \frac{1}{t} \times t \geq 1$$

Rule 1 holds. ■

4 Experimental Results

In this section, we evaluate the performance of the OPMET algorithm using the following three criteria:

1. Number of nodes generated: It represents the total number of enzyme combinations tested to complete the search. The lesser the number of nodes generated, the better the performance of the method.

2. Optimal node rank: This indicates the number of nodes explored before the method arrives at the optimal solution. A small optimal node rank indicates that the strategy efficiently tested the optimal combination as soon as the search started.

3. Execution time: This indicates the total time taken by the method to finish the search and conclude that it found an optimal solution.

We implemented the OPMET algorithm with a random enzyme ordering, Sta-

tic OPMET and Dynamic OPMET (see Subsection 3.2). We implemented two filtering strategies, Non-target filter and Target filter (see Subsection 3.3), to provide pruning capabilities to the OPMET algorithm. We compare the methods we implemented to a hypothetical exhaustive search which enumerates and tests all possible combinations of enzymes (2^n, where n is the number of enzymes).

We extracted the metabolic network information of Escherichia Coli from KEGG [18, 19] (`ftp://ftp.genome.jp/pub/kegg/pathways/eco/`). The metabolic network in KEGG has been divided into smaller networks according to their specific functions. We chose six of these networks for our experiments, based on the number of enzymes. We devised a labeling scheme for the networks which begins with 'N' and is followed by the number of enzymes in the network. For instance, 'N20' indicates a network with 20 enzymes. Table 1 shows the metabolic networks chosen, along with their identifiers and the number of compounds (C) and reactions (R). 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 ran our experiments on an Intel Pentium 4 processor with 2.8 GHz clock speed and 1-GB main memory running Linux operating system.

4.1 Evaluation of OPMET prioritization strategies

In this section, our goal is to evaluate the performance of our enzyme prioritization strategies. Here, we compare our static and dynamic OPMET algorithms with a random ordering of enzymes and an exhaustive search. We do not include

Table 1: Metabolic networks from KEGG with identifier (Id). C and R denote the number of compounds and reactions respectively.

Id	Metabolic Network	C	R
N14	Citrate or TCA cycle	21	35
N17	Galactose	35	50
N20	Pentose phosphate	26	37
N24	Glycerolipid	32	49
N28	Glycine, serine and threonine	36	46
N32	Pyruvate	21	51

Table 2: Average number of nodes generated (#N) and Optimal Node Rank (ONR) of the different ordering strategies, namely, Random Order (RO), Static OPMET (SO) and Dynamic OPMET (DO) compared with hypothetical exhaustive search (ES).

Id		ES	RO	SO	DO
N14	#N	16384	4190	3859	3273
	ONR		1220	3.65	3.77
N17	#N	131072	58379	46411	38973
	ONR		22303	140.87	2.54
N20	#N	1048576	147605	82643	78257
	ONR		100845	440.48	11.36

our filtering strategies here as the goal is to select the best possible enzyme ordering. We have shown the results only up to a network of size 20 enzymes. This is because, beyond this, the search space grows rapidly, necessitating the application of filtering strategies.

Table 2 shows the average number of nodes generated and the average optimal node rank of the random ordering, static OPMET and dynamic OPMET, as compared to an exhaustive search. The average number of nodes results show that Dynamic OPMET is the best strategy for all the tested networks. Dynamic OP-

MET generates only 7 % of the nodes compared to an exhaustive search for N20. Whereas a random enzyme ordering generates 14 % of the nodes compared to an exhaustive search for the same network. Dynamic OPMET generates 30 % of the nodes for N17. This is still lesser than static priority (35 %) and random ordering (45 %). Larger values for N17 are expected because the number of reactions and compounds of this network is much larger than the other networks, resulting in more interactions in the network (see Table 1).

The optimal node rank results show that Dynamic OPMET has the lowest optimal node rank on average. On average, it arrives at the optimal solution within the generation of 0.008 % of the number of nodes possible in an exhaustive search. This is significantly better than the random ordering, which arrives at the optimal solution within the generation of 11 % of the nodes on average and Static OPMET, which arrives at the optimal solution within generation of 0.05 % of the nodes on average.

We conclude that Dynamic OPMET is the best enzyme prioritization strategy and hence use this strategy in the rest of the experiments.

4.2 Evaluation of OPMET filtering strategies

Though Dynamic OPMET arrives at the optimal solution early, it tests a large number of combinations before concluding that the solution found is optimal. In this section, we evaluate how much our filtering strategies reduce the search space. Figure 5 shows the results for Dynamic OPMET with target, non-target filters and a combination of both filters compared to Dynamic OPMET without filters.

Figure 5(a) shows the average number of nodes generated by the four methods. The combined filters show the best pruning. On an average, the combined filters prune 91.5 % of the nodes generated in the method without filters. We also see that most of this benefit is obtained from the target filter (it filters 91.4 % of the nodes generated by the method without filters). The combined filters generate only 12700 nodes for N24 (0.004 % of an exhaustive search of 2.68×10^8 combinations).

Figure 5(b) shows the average execution time. We see that the average execution time shows a trend similar to Figure 5(a). This establishes that the execution time is proportional to the number of nodes generated. The combined filters effectively reduce the execution time to one-tenth of the method without filters on average.

Figure 5(c) shows the average optimal node rank. All the methods have the same optimal node rank for networks except N24. This suggests that Dynamic OPMET yielded the optimal solution as early as possible for these networks. The results for N24 show that filtering strategies can also lead to advancement in finding the optimal solution. For N24, Target filter arrives at the optimal solution 99 % earlier and the combined filters arrive at the optimal solution 99.9 % earlier than the method without filters (the additional 0.9 % improvement is obtained from the non-target filter).

We observe that the target filter is more efficient than the non-target filter. We conclude that Dynamic OPMET with combined filters is the most effective strategy for pruning vast amounts of the search space.

4.3 Scalability in number of targets

In this section, we show the scaling of performance of our method with target size. The target sets compared are of sizes one, two and four. We evaluate Dynamic OPMET with combined filters in this subsection, as it is the most effective combination. Figure 6 shows the results for networks of different sizes.

Figure 6(a) shows the average number of nodes generated for the three target sets. The network topology determines how the target set size affects the number of nodes generated. The target compound set can contain highly correlated compounds (deleted by the inhibition of almost the same set of enzymes) or unrelated compounds (located in different parts of the network). If the target set consists of correlated compounds, an increase in the target set size decreases the average number of nodes generated. This can be seen for N14, N17, N20 and N24 (from 2 to 4 target compound queries). The number of nodes generated for the two-target sets is 62 % lesser than single target sets on an average. For the four-target sets, this value is 75 % lesser.

On the other hand, if the target set consists of unrelated compounds, an increase in the target set size increases the average number of nodes generated. The average number of nodes increases by 1.2 times when we go from single target to two-target sets and by 2.2 times when we go from the two-target to the four-target sets.

Figure 6(b) shows the average execution time for the three target sets. We see that the average execution time shows a trend similar to Figure 6(a). It indicates

that the execution time is proportional to the number of nodes generated.

Figure 6(c) shows that the average optimal node rank increases sub linearly with the number of target compounds. On an average, The two-target queries arrive at the optimal solution after generation of 1.8 times more nodes than the single target queries. Similarly, the four-target queries generate 3.8 times more nodes than the single target queries before arriving at the optimal solution. This suggests that as the target set size increases, the number of enzyme combinations that need to be tested before we find the optimal solution increases.

5 Related Work

The classical drug discovery approach involves incorporating large number of hypothetical targets into in-vitro or cell-based assays and performing automated high throughput screening (HTS) of vast chemical compound libraries [26, 5]. The developments in the post-genomic era promise to provide rational drug-design methods and reduction of serious side-effects [6, 4, 3]. The advances in bioinformatics have led to the concept of *reverse pharmacology* [25]. The first step in this approach is the identification of protein targets that may be critical intervention points in a disease process [22, 1]. The reverse approach is driven by the mechanistic basis of the disease and hence is expected to be more efficient than the classical approach [25].

Rapid identification of optimal 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 [16, 11]. These studies have revealed important properties of these networks [8, 2, 14]. The potential of an enzyme to be an effective drug target is considered to be related to its essentiality in the corresponding metabolic network [12]. Lemke et. al proposed the measure *enzyme damage* as an indicator of enzyme essentiality [13, 15]. Recently, a computational approach to prioritize potential drug targets for antimalarial drugs was developed [17]. A chokepoint analysis of *P.falciparcum* was performed to identify essential enzymes which are potential drug targets. The protein kinase targets that hold potential to reduce insulin resistance and normalize blood glucose are discussed in [21]. These studies show the effectiveness of computational techniques in reverse pharmacological approaches.

A combination of microarray time-course data and gene-knockout data is used to provide an accurate gene network to study the effects of a chemical compound on the network [10]. An investigation of metabolite essentiality is carried out with the help of stoichiometric analysis in [9]. These approaches reveal the importance of studying the role of compounds (metabolites) during the pursuit of computational solutions to pharmacological problems.

6 Conclusion

Efficient computational methods are required to identify the optimal enzyme-combination (i.e., drug targets) whose inhibition will achieve the required effect of eliminating a given target set of compounds while incurring minimal side-effects. In this paper, we formulated the optimal enzyme-combination identification prob-

lem as an optimization problem on metabolic networks. We defined a graph based computational model of the network that encapsulates the impact of enzymes onto compounds. We proposed OPMET, a branch-and-bound algorithm to explore the search space. We developed a cost model and two enzyme prioritization strategies, Static OPMET and Dynamic OPMET based on it. We also developed two filtering strategies to prune the search space while still guaranteeing an optimal solution. The filters compute an upper bound to the number of target compounds deleted and a lower bound to the side-effect respectively.

Our experiments on E.Coli metabolic network show that our methods reduce the total search time by several orders of magnitude as compared to the exhaustive search. The optimal solution is reached by Dynamic OPMET within the exploration of 0.005 % of the total search space on average, proving that our methods are effective in approximating the impact of enzymes on compounds. Dynamic OPMET with combined filters pruned 91.6 % of the search space on average.

In this paper, we have developed a foundation for developing effective solutions to pharmacological problems, through analysis of metabolic networks. We are currently working to improve the accuracy of our network and cost models in modeling the metabolic network's behavior. We are also developing methods which will efficiently provide solutions to the same problem for larger metabolic networks.

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):1543–1547, Feb 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] J Drews. Drug Discovery: A Historical Perspective. *Science*, 287(5460):1960–1964, Mar 2000.

[6] Davidov et. al. Advancing drug discovery through systems biology. *Drug Discovery Today*, 8(4):175–183, Feb 2003.

[7] 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.

[8] Hatzimanikatis et. al. Metabolic networks: enzyme function and metabolite structure. *Current Opinion in Structural Biology*, (14):300–306, 2004.

[9] Imielinski et. al. Investigating metabolite essentiality through genome scale analysis of E. coli production capabilities. *Bioinformatics*, Jan 2005.

[10] Imoto et. al. Computational Strategy for Discovering Druggable Gene Networks from Genome-Wide RNA Expression Profiles. In *PSB 2006 Online Proceedings*, 2006.

[11] Jeong et. al. The large-scale organization of metabolic networks. *letters to NATURE*, 407:651–654, Oct 2000.

[12] Jeong et. al. Prediction of Protein Essentiality Based on Genomic Data. *ComPlexUs*, 1:19–28, 2003.

[13] Lemke et. al. Essentiality and damage in metabolic networks. *Bioinformatics*, 20(1):115–119, Jan 2004.

[14] Ma et. al. Decomposition of metabolic network into functional modules based on the global connectivity structure of reaction graph. *Bioinformatics*, 20(12):1870–1876, 2004.

[15] Mombach et. al. Bioinformatics analysis of mycoplasma metabolism: Important enzymes, metabolic similarities, and redundancy. *Computers in Biology and Medicine*, 2005.

[16] Teichmann et. al. The Evolution and Structural Anatomy of the Small Molecule Metabolic Pathways in Escherichia coli. *JMB*, 311:693–708, 2001.

[17] Yeh et. al. Computational Analysis of Plasmodium falciparum Metabolism: Organizing Genomic Information to Facilitate Drug Discovery. *Genome Research*, 14:917–924, 2004.

[18] M Kanehisa. A database for post-genome analysis. *Trends in Genetics*, 13(9):375–376, Sep 1997.

[19] M Kanehisa and S Goto. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 28(1):27–30, Jan 2000.

[20] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[21] C. M Rondinone. Serine Kinases as New Drug Targets for the Treatment of Type 2 Diabetes . *Current Medicinal Chemistry - Immunology, Endocrine and Metabolic Agents*, 5(6):529–536, Dec 2005.

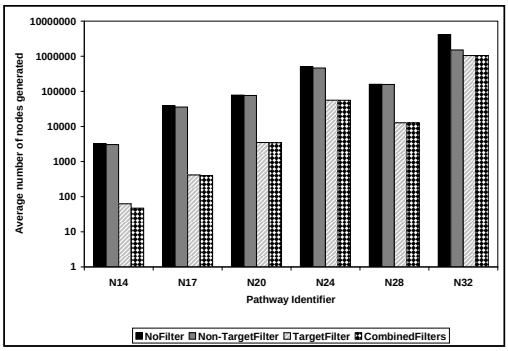
[22] C Smith. Hitting the target. *Nature*, 422:341–347, Mar 2003.

[23] R. Somogyi and C.A. Sniegoski. Modeling the complexity of genetic networks: understanding multigene and pleiotropic regulation. *Complexity*, 1:45–63, 1996.

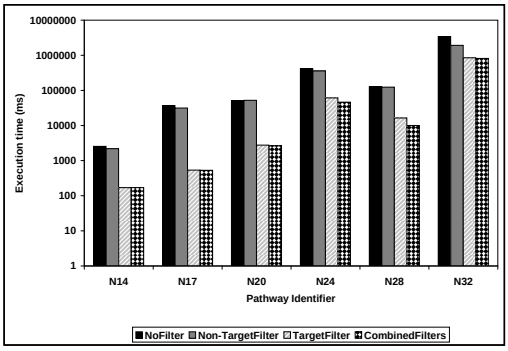
[24] R. Surtees and N. Blau. The neurochemistry of phenylketonuria. *European Journal of Pediatrics*, 159:109–13, 2000.

[25] Takenaka T. Classical vs reverse pharmacology in drug discovery. *BJU International*, 88(2):7–10, Sep 2001.

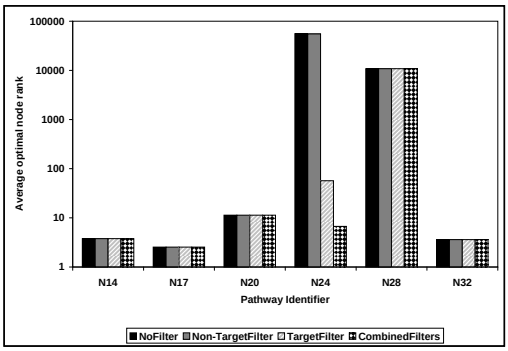
[26] G Wess. How to escape the bottleneck of medicinal chemistry. *Drug Discovery Today*, 7(10):533–535, May 2002.



(a) Average number of nodes generated

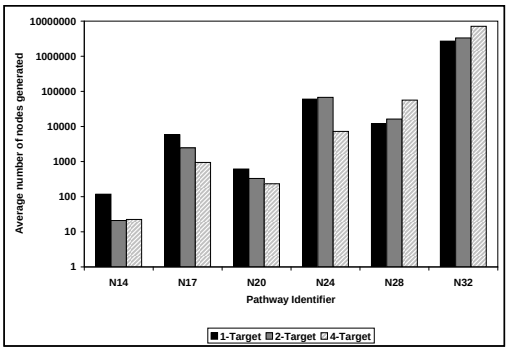


(b) Average execution time in milliseconds

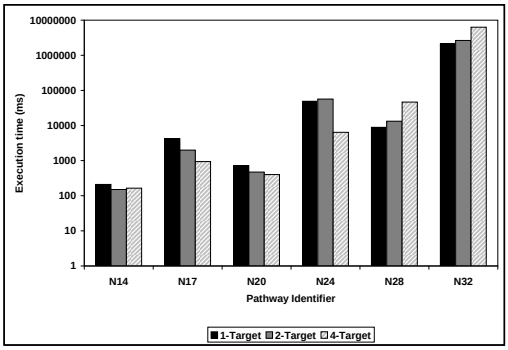


(c) Average optimal node rank

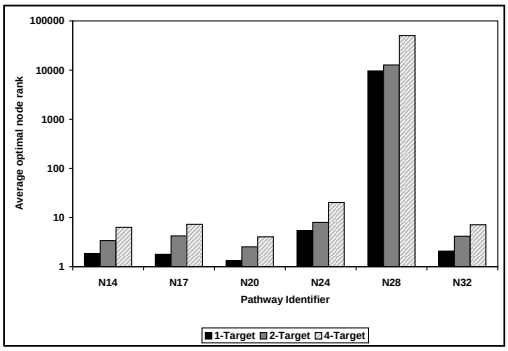
Figure 5: Comparison of OPMET filtering strategies



(a) Average number of nodes generated



(b) Average execution time in milliseconds



(c) Average optimal node rank

Figure 6: Scaling of Dynamic OPMET with combined filters with respect to the target compound set size