BIOINFORMATICS

Vol. 00 no. 00
Pages 1–11

RINQ: Reference-based Indexing for Network Queries

Günhan Gülsoy , Tamer Kahveci

Computer and Information Sciences and Engineering, University of Florida, Gainesville, FL 32611
{ggulsoy,tamer}@cise.ufl.edu

ABSTRACT

We consider the problem of similarity queries in biological network databases. Given a database of networks, similarity query returns all the database networks whose similarity (i.e., alignment score) to a given query network is at least a specified similarity cutoff value. Alignment of two networks is a very costly operation, which makes exhaustive comparison of all the database networks with a query impractical. To tackle this problem, we develop a novel indexing method, named RINQ (**R**eference-based **I**ndexing for Biological **N**etwork **Q**ueries). Our method uses a set of reference networks to eliminate a large portion of the database quickly for each query. A reference network is a small biological network. We precompute and store the alignments of all the references with all the database networks. When our database is queried, we align the query network with all the reference networks. Using these alignments, we calculate a lower bound and an approximate upper bound to the alignment score of each database network with the query network. With the help of upper and lower bounds, we eliminate the majority of the database networks without aligning them to the query network. We also quickly identify a small portion of these as guaranteed to be similar to the query. We perform pairwise alignment only for the remaining networks. We also propose a supervised method to pick references that have a large chance of filtering the unpromising database networks. Extensive experimental evaluation suggests that (1) our method reduced the running time of a single query on a database of around 300 networks from over two days to only eight hours; (2) our method outperformed the state of the art method Closure Tree and SAGA by a factor of three or more; (3) our method successfully identified statistically and biologically significant relationships across networks and organisms.

Keywords: Biological network databases; indexing; network comparison

1 INTRODUCTION

With the advances in biotechnology and high throughput computing, the amount of data describing the interactions among molecules has increased greatly in recent years. Depending on the types of interactions, this data is expressed through various models such as metabolic (Francke *et al.*, 2005), gene regulatory (Levine and Davidson, 2005) or protein interaction (Giot and Bader et al, 2003) networks. The term pathway is also used in the literature to describe a part of them. In this paper, we will use the term *biological network* to describe a collection of these interactions. Understanding biological networks has been one of the main goals of biological sciences as they contain key information regarding how organisms work. To achieve this goal, biological networks are analyzed in a number of ways. One of them, comparison based analysis, identifies similar parts of two biological networks by aligning them. Such analysis has been successfully used for finding functional annotations (Clemente *et al.*, 2006), identifying drug targets (Sridhar *et al.*, 2007), reconstructing metabolic networks from newly sequenced genome (Francke *et al.*, 2005), and building phylogenetic trees (Clemente *et al.*, 2005).

Aligning two biological networks is a computationally challenging problem. Existing methods often map this problem to subgraph isomorphism problem. For this problem, there are no known polynomial time algorithms which guarantee to produce the optimal alignment. A number of methods aim to find approximate (Dost *et al.*, 2008; Pinter *et al.*, 2005; Shlomi *et al.*, 2006) or heuristic (Ay and Kahveci, 2010; Ay *et al.*, 2009; Liao *et al.*, 2009) results in practical time. However, they still require a significant amount of time.

Computational difficulty of biological network alignment becomes apparent when we need to align a query network with a database of biological networks. To test this, we downloaded 297 networks from KEGG (Ogata *et al.*, 1999). These networks had 16 to 120 genes (i.e., nodes) and 17 to 165 edges. We aligned the networks in the dataset with a set of 50 query networks using the QNet algorithm (Dost *et al.*, 2008). Each of these query networks had seven nodes. Average processing time for a single query network was over two days on a single processor. The same experiment for eight node queries took more than a week for a single query network. In this paper, we aim to align a query network with a database of networks efficiently. Formal definition of our problem is as follows:

Problem definition: *Assume that we have a database of* n *biological networks denoted by* $D = \{d_1, d_2, \ldots, d_n\}$*, where* d_i *is the ith network in our database. Also, we are given a query network denoted by* q*. Let us denote the alignment score between* q *and* d_i ($d_i \in D$) *with* $sim(q, d_i)$*. Given a similarity cutoff* ϵ*,* similarity search *returns all the database networks* d_i *that satisfy* $sim(q, d_i) \geq \epsilon$*. We aim to reduce the processing time of similarity searches in biological network databases to a practical level.* □

A number of variants of the problem of aligning two networks have been considered in the literature (Ay and Kahveci, 2010; Dost *et al.*, 2008; Kelley *et al.*, 2004; Liao *et al.*, 2009; Shlomi *et al.*, 2006). However, there has been a limited number of studies on similarity searches in network databases (Giugno and Shasha, 2002; He and Singh, 2006; Mongiovi *et al.*, 2010; Yan *et al.*, 2004). Although these methods work well for specific types of network databases, they are not well suited for biological network databases which is the problem considered in this paper. This is mainly because they often use an overly simplified similarity measure for network alignment. When used with costly biological network alignment tools, the time complexity of these indexing methods become impractical. We elaborate on previous works in Section 2.

Contributions: In this paper, we develop a reference-based indexing method to answer similarity queries in biological network databases efficiently. We call this method **R**eference-based **I**ndexing for Biological **N**etwork **Q**ueries (RINQ). The main advantage of this method is its independence of the pairwise network alignment algorithm the database employs. Briefly, RINQ works as follows. First, we choose a small set of networks to use as *reference networks*. We align all of the reference networks with the networks in our database, and store all the alignment mappings and scores. These steps are performed offline. After these two steps, our database is ready to accept queries. Given a query q and a similarity cutoff ϵ, instead of aligning q with all the database entries, we align it with the reference networks. Using these alignments and the precomputed alignments between reference and database networks, we compute a lower bound and an approximate upper bound for $sim(q, d_i)$ for each database network d_i. We will refer to these upper and lower bounds as $UB(q, d_i)$ and $LB(q, d_i)$ respectively. Computing upper and lower bounds take much less time than aligning the query network with the database networks. Depending on the values of the upper and lower bounds, we may encounter one of the following three cases.

- Case 1: (Filter Set) $UB(q, d_i) < \epsilon$
- Case 2: (Result Set) $\epsilon \leq LB(q, d_i)$
- Case 3: (Twilight Zone) $LB(q, d_i) < \epsilon \leq UB(q, d_i)$

We can interpret these cases as follows. The first case means that $sim(q, d_i) < \epsilon$. We filter such database entries. Second case implies $sim(q, d_i) > \epsilon$, which is what we are looking for. We can directly add such d_i to our result set. For the database networks d_i that fall into the third case, the value of $sim(q, d_i)$ is unclear. We align q with such d_i to find their actual similarity value. In other words, we perform the costly network alignment algorithm only for the networks that fall into the third case. Our experiments suggest that

1. RINQ filters up to 80 percent of the database entries depending on the given similarity cutoff and the query.
2. RINQ is significantly faster and more accurate than the existing methods including SAGA (Tian *et al.*, 2007) and CTree (He and Singh, 2006).
3. The alignments returned by RINQ are statistically significant and biologically relevant.

Rest of the paper is organized as follows. Section 2 discusses the previous works related to our problem. Section 3 elaborates on RINQ. Section 4 presents our experimental results. Section 5 briefly concludes the paper.

2 RELATED WORK

Most of the existing work on comparing biological networks focus on alignment of a pair of networks. This problem is the fundamental building block of the similarity search problem considered in this paper. However, aligning even a single pair of networks is a computationally difficult task. To alleviate this difficulty a number of methods in the literature restricted alignments to specific topologies or network sizes. For example, PathBLAST (Kelley *et al.*, 2004) and QPath (Shlomi *et al.*, 2006) align linear pathways with protein interaction networks. Pinter $et\ al.$ devised an algorithm which can align queries of tree networks with a multi source tree network (Pinter *et al.*, 2005). QNet uses a color-coding algorithm to align trees or bounded treewidth networks with any network with provable confidence values (Dost *et al.*, 2008). To evaluate the performance of QNet, we aligned a set of seven node query networks with a database of gene regulatory networks using our implementation of the QNet algorithm. We downloaded all the gene regulatory networks from KEGG (Ogata *et al.*, 1999) that had more than 15 nodes in our local database. There were 297 such networks (See Tables 3, 4 in Appendix II). When we set the confidence value to 99%, the average running time to align a single query network with all database networks exhaustively took over two days. Even after we reduced the confidence to 90%, the running time for exhaustive search was still one day. A number of other approaches exist to speed up for pairwise network alignment problem. NetMatch (Ferro *et al.*, 2007) and NetGrep (Banks *et al.*, 2008) allow user to define queries with specific node features. Then they use fast approximate algorithms to match the user defined queries. Heuristic methods aim to solve the alignment problem faster without providing a confidence value (Ay and Kahveci, 2010; Bruckner *et al.*, 2009; Liao *et al.*, 2009). However, despite all these efforts, the running time still remains to be a bottleneck.

Several indexing methods for similarity searches in graph databases exist. Majority of these methods can be classified as *feature based indexing* methods. These methods start by picking specific features of the networks for filtering purposes. GraphGrep chooses paths as index feature (Giugno and Shasha, 2002). gIndex uses frequent subgraphs for the same purpose (Yan *et al.*, 2004). SAGA uses fragments of database networks and tries to combine them together to find larger matches (Tian *et al.*, 2007). These methods focus on finding exact matches of a given query in the database. SIGMA is another feature based indexing method which concentrates on the problem of inexact matching in graph databases (Mongiovi *et al.*, 2010). Another method for graph database indexing is Closure-Tree (CTree) (He and Singh, 2006). Closure-Tree organizes the networks in the database using a binary tree structure. It places each database network at a different leaf node in this tree. Each internal node in this tree is a hypothetical network that is obtained by aligning the two networks corresponding to its children nodes. An interesting property of the CTree is the following. The score of the alignment of any query network with an internal node is at least as much as that with a leaf node rooted at that internal node. Following from this, given a query network, closure tree algorithm starts aligning query to the root node. It then proceeds to the children nodes. It prunes an entire subtree rooted at an internal node, if the alignment to that internal node has a score less than the given cutoff.

Existing graph database indexing methods work well when each database network is small or the underlying similarity measure is not expensive. However, these two properties do not hold for biological network databases such as metabolic and gene regulatory networks. As a result, when they are applied to these networks, either the index construction time becomes impractical or the amount of performance gained becomes too small. For instance, when a costly dynamic programming method, such as QNet (Dost *et al.*, 2008), is used to summarize sets of networks accurately in the CTree method, computing the summary for even a single internal node can take more than one month for networks with more than 12 nodes. *On the other hand, the method we develop in this paper allows querying databases with networks of arbitrarily large number of nodes.* We provide an experimental comparison of our method with SAGA and CTree in Section 4.3.

3 ALGORITHM

In this section, we explain the indexing method we developed for querying biological network databases. We first describe the notation we use in the rest of this paper. We denote the database of n biological networks with $D = \{d_1, d_2, ..., d_n\}$. Here, d_i represents the ith network in the database. As we will explain later, our algorithm uses a set of reference networks. We denote the set of m candidate references with $C = \{c_1, c_2, ..., c_m\}$. In this representation, each c_i corresponds to a candidate reference network. We denote the actual set of references with $R = \{r_1, r_2, ..., r_k\}$, where $R \subseteq C$. To simplify our notation, we will drop the subscript and use d, r or c to represent d_i, r_i or c_i respectively whenever possible. We will represent a query network with q. We use $q[j]$, $c[j]$, $r[j]$ and $d[j]$ to refer to the jth node in respective networks.

An alignment is a mapping between the nodes of two networks. Let us represent the mapping between q and d with β. Then, $\beta(q[i]) = d[j]$ means that $q[i]$ is aligned to $d[j]$. Note that all these mappings are one to one and symmetric. In other words, if $\beta(q[i]) = d[j]$, then $\beta(d[j]) = q[i]$. When a node $q[i]$ is not aligned to any node in d (i.e., insertions or deletions), we will use the notation $\beta(q[i]) = \emptyset$. The similarity score between two nodes $q[i]$ and $d[j]$ is represented as $sim(q[i], d[j])$. Let E_d and E_q be the set of edges (i.e., interactions) of d and q respectively. Let us denote the function that returns the weight of an edge in E_d with EW_d. Given a mapping β between q and d, the individual node similarities between q and d, we calculate the similarity score $sim(q, d)$ as:

$$\begin{aligned} sim(q,d) &= \sum_{\beta(q[i]) \neq \emptyset} sim(q[i], \beta(q[i])) \\ &+ \sum_{\beta(q[i]) = \emptyset} InDel\ Penalty \\ &+ \sum_{\substack{(\beta(q[i]),\, \beta(q[j])) \in E_d \\ \text{AND } (q[i],\, q[j]) \in E_q}} EW_d(\beta(q[i]), \beta(q[j])). \end{aligned}$$

The organization of the rest of this section is as follows. In Section 3.1, we provide an overview of reference-based indexing. We elaborate on our reference selection method in Section 3.2. Finally, in Section 3.3, we explain the calculation of the lower and upper bounds we use for RINQ.

3.1 An overview of our reference-based indexing

RINQ has two major steps, namely index creation and query processing. We create index once as a preprocessing step in two phases. In the first phase, we create a large set of candidate references from the database networks. In the second phase, we pick a subset of these candidates as the actual reference set by considering their performance over a set of training queries. In our index, we store the alignments between all the reference and database network pairs.

Once the index is created, we are ready to answer similarity queries on our database. Given a query network q, we align q with the reference networks in R rather than the database networks. Usually, the cost of these alignments is negligible since R contains much fewer and smaller networks than D. For each $d_j \in D$, we compute a lower bound and an approximate upper bound to the similarity score between q and d_j. We do this by using the alignments of the references to q and d_j. We denote these lower and upper bounds by $LB(q, d_j)$ and $UB(q, d_j)$ respectively.

Using the lower and upper bound values, we classify each d_j in one of the following three sets: *Filter Set*, *Result Set*, or *Twilight Zone*. If $UB(q, d_j) < \epsilon$, the alignment score between q and d_j cannot be larger than ϵ. We put such d_j in *Filter Set*. We eliminate the networks in the filter set without further calculation. If $\epsilon \leq LB(q, d_j)$, it means d_j has an alignment score which is definitely larger than ϵ. So d_j should be in the set we return as the result of our query. We place such d_j in the *Result Set*. Finally, if $LB(q, d_j) < \epsilon \leq UB(q, d_j)$, the bounds do not imply anything about the network being in the result set or the filter set. We place such d_j in the *Twilight Zone*. For the entries in the twilight zone, we need to calculate the actual alignment scores to decide if we can place them in the result set. *Since aligning two networks is a costly operation, we would like to have as few networks in the twilight zone as possible.*

We discuss how we create the reference set in Section 3.2. We defer the discussion on the lower and upper bound computations using references to Section 3.3.

3.2 Reference selection

The success of our method depends greatly on the reference set. Ideally, a reference set should contain reference networks from which we can construct a network that is similar to the given query network. This section describes how we choose our reference networks for our database with the help of a training query set. Briefly, we select references in two phases. In the first phase, we quickly generate a set of promising references. In the second phase, we carefully examine these candidates and choose a subset of them that has maximal performance on the training queries.

3.2.1 Phase I: Creating the candidate reference set. Ideally the reference set in our database should satisfy the following properties:

Property 1. (SMALL) Each reference network should have a small number of nodes.

Property 2. (COMPREHENSIVE) At least one reference should align well with any database network.

Property 3. (NON-REDUNDANT) Each reference network should differ from the rest significantly.

The first property above ensures that the query aligns with each reference quickly. The second property ensures that it is possible to find tight lower and upper bounds for any query that aligns well with a reference network. If two references are highly similar, then the lower and upper bounds computed through them will be similar regardless of the query network. Thus one of them is sufficient to compute the bounds. The third property avoids such redundancy.

In order to satisfy the first rule we set the size (i.e, the number of nodes) of each reference network to that of the largest query allowed. Let us denote this number with T. We use $T = 6$, 7 or 8 in our experiments. Using small T ensures Property 1 described above. Following from the rules above, we create a potentially large set of candidate references from the database networks iteratively. Each iteration consists of two steps.

Step 1. We pick a database network randomly. We choose a random node from that network as the seed. We then grow that seed by including one of the neighboring nodes to the seed randomly, until the seed contains T nodes.

Step 2. We align the current seed with rest of the candidate reference networks we created so far. If any of the alignment scores is greater than a predefined cutoff, it means that the current seed is redundant.

Table 1. The distribution of the number of database networks to different sets for a given query. The networks in the partition *True* are the ones that need to be returned for the query. The networks in the partition *False* are the ones that are not similar to the query. The networks are also divided into three sets, *Result Set*, *Twilight Zone* and *Filter Set*. All the N_i inside the partitions represent the number of network in the corresponding partition.

	Result Set	Twilight Zone	Filter Set
True	N_1	N_2	N_3
False	0	N_4	N_5

Therefore, we discard the current seed. Otherwise, we include the current seed in the candidate reference set.

Steps 1 and 2 eliminates redundancy from the candidate reference network set, therefore enforcing Property 3. The iterations continue until we are unsuccessful at creating a new candidate reference network, hence fulfilling Property 2.

3.2.2 Phase II: Creating the final reference set. Recall from Section 3.1 that we align a given query network with all the reference networks in R first. In order to perform this comparison quickly, it is desirable to have a small number of references in R. The amount of time we would like to spend for filtering the database determines the size of R. We will use at most 100 references in our experiments. We choose our actual reference set as a subset of the candidate references. A naive solution would be to pick the reference networks randomly from the candidate reference set. This simple strategy may produce a good reference set since the candidate set already satisfies the three properties listed at the beginning of this section. Here, we develop an intelligent strategy that optimizes the reference selection further. This strategy learns the *accuracy* of the references by training over a set of queries.

At this point, we take small detour to explain how we compute the accuracy of our references for a query. As we explained in Section 3.1, our references place each database network into one of the three sets, namely Result Set, Filter Set and Twilight Zone. Among these, the last two may contain true or false results (i.e., a database network is a true result if its alignment to the query has a score greater than the given cutoff). Table 1 shows the decomposition and the number of networks in each set. Result set contains only true results as our lower bound is guaranteed to be at most the actual score. The filter set on the other hand may contain true results, as well as false results. Our method fails to find the true results in this set if there are any. We compute the accuracy of our method as the ratio of true results that our method can return. Formally this number is $\frac{N_1+N_2}{N_1+N_2+N_3}$. This metric is also known as *recall* in the literature.

Algorithm 1 iteratively tries to find the reference set which yields the highest accuracy. We do this with the help of a set of training query networks. We avoid a purely combinatorial approach because the number of possible reference sets is very high. Instead, we develop a hill climbing approach. Algorithm 1 describes this method in detail. We start by randomly choosing a reference set. Then, at each iteration, we replace one reference network in R with another in $C - R$ to improve the accuracy of our index for the training queries. To do this, we first eliminate the reference from the current reference set whose removal drops the accuracy the least. Next, among all the remaining candidate references, we find the one whose inclusion to the current reference set increases the accuracy the most. If these two changes improve the overall accuracy we accept them. We repeat these iterations until there is no candidate reference which can improve the overall accuracy.

```
Algorithm 1: Create reference set $R$
Align all the networks in $Q$ and $C$ with the networks in $D$
Remove $k$ networks from $C$ randomly and add them to $R$
repeat
  Set Current Accuracy to the accuracy of $R$
  Remove $r_i$ from $R$ where accuracy of $R - \{r_i\}$ is maximum
  for all $c_j$ in $C$ do
    Calculate accuracy using $R \cup \{c_j\}$
  end for
  if The resulting accuracy is better than the current one then
    Add $c_j$ to $R$ for which the accuracy of $R \cup \{c_j\}$ is largest
    Remove $c_j$ from $C$
  else
    Insert $r_i$ back in $R$ and Break the loop.
  end if
until Accuracy cannot be improved anymore or $C$ is empty
```

3.3 Computing the bounds

For a query network q, we calculate $UB(q, d)$ and $LB(q, d)$ for each $d \in D$ with the help of references. When we create the reference set as described in Section 3.2, we also record the alignment of each (r_i, d) pair ($r_i \in R$). For each (q, d) pair, we compute $UB(q, d)$ and $LB(q, d)$ using the precomputed alignments of each r_i with d, and the alignments of each r_i with q. In Section 3.3.1, we present an exact lower bound calculation technique. Then in Section 3.3.2, we discuss an approximate upper bound calculation method.

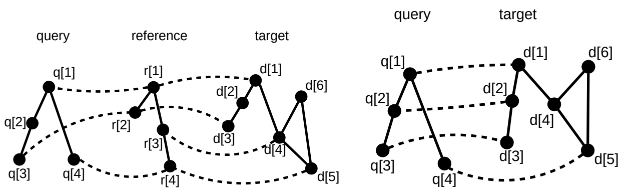
In order to simplify the explanation of bound calculations, we denote an alignment of q and r_i with a relation ψ_i. We will use the relation δ_i to represent the alignment between r_i and d.

3.3.1 Lower bound calculation In order to provide a tight lower bound for a given query and database network pair, we use each reference independently. Using each reference network r_i, we calculate a different lower bound value $LB_i(q, d)$. After calculating k different lower bounds, we choose the largest one (i.e., $LB(q, d) = \max_i\{LB_i(q, d)\}$) as the lower bound.

Given q, r_i and d, we compute $LB_i(q, d)$ in two phases. In the first phase, we map a subset of the nodes of q to those of d from their existing alignments with r_i. In the second phase, we map the remaining nodes if possible and calculate final alignment score.

Phase I. Using the representations for alignments between (q, r_i) and (r_i, d) pairs, our indirect alignment ϕ_i between q and d through r_i is simply the composition of relations δ_i and ψ_i. In other words, it is the relation $\phi_i(q[j]) = \delta_i(\psi_i(q[j]))$. For example, in Figure 1(a), $q[1]$ and $r_i[1]$ are aligned (i.e., $\psi_i(q[1]) = r_i[1]$). Also $r[1]$ and $d[1]$ are aligned (i.e., $\delta_i(r_i[1]) = d[1]$). From these two, in Figure 1(b) we align $q[1]$ with $d[1]$ (i.e., $\phi_i(q[1]) = d[1]$). Notice that ϕ_i maps each node of q (or d) to at most one node in d (or q). This is because both δ_i and ψ_i map one node of q and d to at most one node in r_i. Also notice that there may be nodes in q or d that are not aligned to any node by ϕ_i.

Phase II. If all the nodes of q are aligned at the end of Phase I, we simply use this alignment to compute $LB_i(q, d)$. Otherwise, there will be unaligned nodes from at least one network. One way to deal with the unaligned nodes is to consider them as insertions or deletions (*indel*) in the alignment of q and d. This, however, is not desirable as each indel will lower the value of $LB_i(q, d)$. We observe that it is possible to align such nodes if there is a matching node in d that is not aligned in Phase I. For example, in Figure 1(a), both $q[2]$



(a) Mappings ψ and δ (b) Indirect alignment ϕ

Fig. 1. Overview of the lower bound calculation. Figure 1(a) shows the alignments of (q,r) and (r,d) pairs. Figure 1(b) shows the resulting indirect alignment. Solid lines represent the edges within a network. Dashed lines represent the mapping between the nodes of two networks.

and $d[2]$ are unaligned with a reference node. Thus, they both are unaligned. In Phase II, we align such nodes with each other. We do this by performing a breadth first traversal on q and d simultaneously. We start the traversal from the root node of q and the node aligned to it in d. Then, we visit every child of this node in q, also visiting the aligned node in d. When visiting a node, we check the shortest path between this node and its parent. If both in q and d there are unaligned nodes, we align such nodes to each other in the order they are traversed. We mark the nodes we cannot align as insertion and deletions. We start the traversal from a pair of nodes that are aligned to each other in Phase I and that are both neighbors to unaligned nodes. Figure 1(b) shows the alignment found after Phase II using the reference alignment in Figure 1(a). Finally, we compute $LB_i(q, d)$ as the similarity score $sim(q, d)$ resulting from the alignment ϕ_i.

3.3.2 Upper bound calculation As we discussed in Section 3.3.1, each reference provides an indirect alignment between q and d. Consider the extreme scenario when one of the references is identical to q. In that case, the indirect alignment of q and d through that reference will be the optimal alignment since d is optimally aligned with each reference in our index. As the similarity between q and a reference increases, we expect that the indirect alignment obtained from that reference will approach to the optimal one. Our upper bound computation strategy follows from this observation. Even when none of the references is identical to q, each one may be similar to a subnetwork of q. Thus, by considering all the references at once, we may be able to reconstruct a subnetwork that is very similar to q. Note that this method is not guaranteed to produce an upper bound. Its success depends on how well the reference set represents q. As we described in Section 3.2, Algorithm 1 chooses the reference set that maximizes the accuracy of our upper bound function. We discuss the observed accuracy of our method later in Section 4.

We calculate $UB(q, d)$ in two phases. In the first phase, we find the set of node pairs from q and d that can be aligned using all the references. In the second phase, we find the highest scoring alignment under this limitation by relaxing the topologies of q and d.

Phase I. For each reference $r_i \in R$, we find an indirect mapping ϕ_i between q and d as we explained in Phase I of Section 3.3.1. We then create a joint mapping ϕ from all indirect mappings as $\phi(d[j]) = \cup_{1 \leq i \leq k}\{\phi_i(d[j])\}$. That is ϕ maps each $d[j]$ to the set of all nodes of q that $d[j]$ is indirectly aligned by at least one reference. Notice that it is possible to have some nodes in d or q that are not mapped to any node by ϕ. This happens when no reference in our reference set aligns with them. This however does not mean that those nodes of q and d are dissimilar. In order to deal with such nodes, we expand ϕ by mapping such such $d[i]$ ($q[j]$) to all the nodes in q (d).

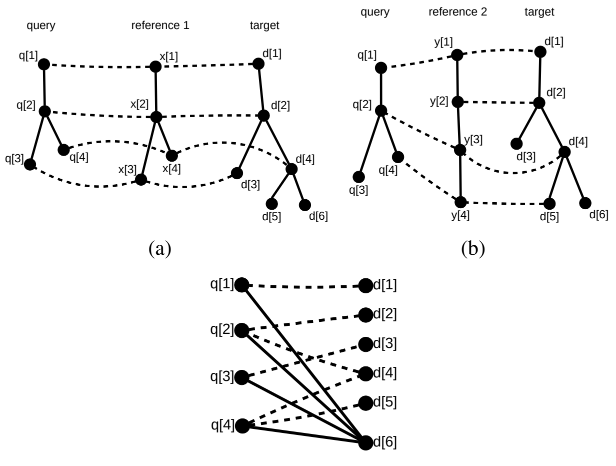


Fig. 2. Overview of the upper bound calculation method. Figures 2(a) and 2(b) show the alignments of q and d with references 1 and 2 respectively. In Figures 2(a) and 2(b), solid lines represent the edges between the nodes of a single network, and dashed lines represent the mappings of two nodes in two different networks. Figure 2(c) shows the bipartite graph resulting from the indirect alignments using two references. In Figure 2(c), all the lines, solid and dashed, represent an element of the relation ψ. Solid lines originating from $d[6]$ show that these elements are added to ψ, since no information about the node $d[6]$ was gathered through references. All the edge weights are omitted in all figures.

Figure 2 depicts this on an example that has two references. In this case, we have two reference networks. Figures 2(a) and 2(b) show the alignments of these references to both q and d. Figure 2(c) shows the relation ϕ based on these two references. Notice that in Figure 2(c), $d[4]$ is mapped to two nodes from q, $q[4]$ and $q[2]$. This is because $d[4]$ is indirectly mapped to $q[4]$ and $q[2]$ through reference 1 and reference 2 respectively. Also, $d[6]$ is mapped to all the nodes in q as it is not aligned to any node in any of the references.

Phase II. In this phase we align q and d using the mapping ϕ obtained in Phase I. Since ϕ is computed from many references jointly, it can map some of the nodes in q to topologically distant nodes of d and vice versa. We ignore the topologies of q and d at this phase. Instead, we consider the mapping ϕ as a weighted bipartite graph. The nodes of q and d are the nodes of this graph. The mappings in ϕ are the edges. The similarity between the nodes that map are the weights of the edges. We compute the upper bound $UB(q, d)$ as the total weight of the maximum weighted bipartite matching between q and d.

4 RESULTS AND DISCUSSION

This section evaluates the performance of RINQ experimentally.

Database. We used networks from the KEGG database (Ogata *et al.*, 1999) in our experiments. We downloaded all the gene regulatory networks that have more than 15 genes (i.e., nodes). In total, there are 297 such networks as of September 2010. In this database, we had 21 different types of networks (such as MAPK signaling pathway) from 46 different organisms. Further details about the size as well as the distribution of the networks to different types and organisms are in Appendix II (See Figures 8, 9 and Tables 3, 4).

Queries. We created three sets of query networks from this dataset by performing random walks on the database networks. These sets contain six, seven and eight node queries respectively. Each query set consisted of 100 query networks. We then randomly split each query set into two disjoint sets, each containing 50 networks. We used one of these sets for training our index, and the other one for testing. All the query networks can be downloaded from our server at `http://bioinformatics.cise.ufl.edu/palRINQ.html`.

Implementation Details. We implemented RINQ in C++. We align two networks using our own implementation of the QNet method (Dost *et al.*, 2008). In order to calculate node similarity scores, we use BLAST (Altschul *et al.*, 1990). We use the minus logarithm of the E-values as the node similarity scores. We carried out experiments with both high and low indel penalties. For high and low indel penalties, we used 1.5 and 0.5 times the largest possible node similarity score respectively.

Methods Compared Against. We compared RINQ against two of the existing filtering methods, SAGA (Tian *et al.*, 2007) and CTree (He and Singh, 2006). These methods did not have their source codes publicly available. Thus, for experiments with SAGA, we used the online SAGA server. The CTree method describes two variations for filtering the internal nodes of the tree, namely maximum weighted bipartite matching and neighbor biased matching. We implemented and tested both of them. We report the results for CTree for only the neighbor biased matching as it performs better than the other alternative.

Evaluation Criteria. We measured the *running time* and *accuracy* as the performance criteria. We measured the performance of our method by changing the values of two variables. These variables are the number of references and query selectivity. In our experiments, the number of references varies from 4 to 100. Query selectivity indicates the percentage of true results for each query. For instance, 4% selectivity means that 4% of the networks in the database are similar to the query according to the given similarity cutoff. We performed experiments with up to 10% selectivity. For each parameter combination, we performed experiments using 50 testing queries. We present the average of 50 queries for each experiment. We used all three sets of queries in our tests. However, due to space restrictions, we only include the tests for seven node queries here. Our other experiments on the other query and reference sets yield similar results.

Test Environment. We performed our tests on Linux servers equipped with dual AMD Opteron dual core processors running at 2.2 GHz, and 3 GB's of main memory.

In Section 4.1 we evaluate the contribution of our reference selection strategy to the performance of RINQ. In Section 4.2, we discuss how our index structure performs for varying index and query parameters. In Section 4.3, we compare the performance of RINQ to SAGA and CTree. Finally, in Section 4.4 we present the statistical and biological significance of our results.

4.1 Effects of reference selection strategy

Recall that we proposed to carefully select a reference set from the candidate set we generated (Section 3.2.1). The first question we need to answer is how each of these steps affect the performance of reference based indexing. In order to do this, we compared our method to two alternatives variants. The first one picks references randomly from the database using random walk. The second one chooses references randomly from the candidate reference set. We conducted two sets of experiments using 4% and 8% query selectivity. In each experiment, we varied the number of references from four to 100 and measured the average accuracy and the running time.

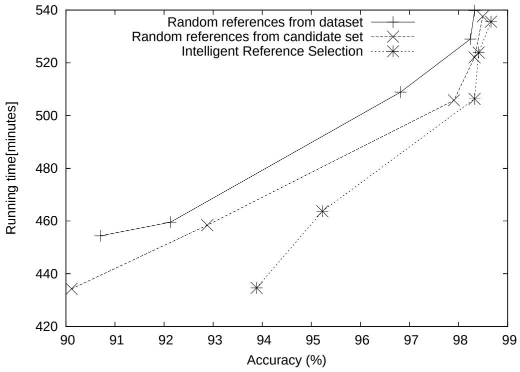


Fig. 3. Running time versus accuracy of our method for different reference selection strategies. Experiments are repeated for different number of references. Average query processing time is presented as running time. Accuracy is calculated over all test queries. Lower running time and higher accuracy indicates better performance. The running time of exhaustive search is over two days (not shown in figure).

Figure 3 shows the results for three reference selection strategies. We also measured the running time of exhaustively searching the database without using our index. The running time of exhaustive search was over two days. The experiments demonstrate that reference based indexing is significantly faster than exhaustive search. It improves the running time by a factor of five over exhaustive search. Creating candidate references by following the three rules (see Section 3.2.1) alone improves the running time by up to 5%. Finally, carefully selecting references using Algorithm 1 results in up to 10% additional improvement. In summary, the results suggest that the reference selection strategy of RINQ indeed helps in improving the performance of our method. It also demonstrates that RINQ makes similarity searches in biological network databases practical.

4.2 Effects of index and query parameters

Earlier in Section 4, we pointed out that two parameters can affect the performance of RINQ, namely the number of references and query selectivity. In our next set of experiments, we evaluate the effects of these parameters on the performance of RINQ. In order to do this, we designed two sets of experiments. In our first set of experiments, we fixed query selectivity to 4% and 8%, and performed database queries with varying number of references for each selectivity value. We carried out the same experiment with RINQ using alignments with both high and low indel penalties. For the second set of experiments, we fixed the number of references and we changed the query selectivity from 3% to 10%. We present the running time and accuracy with respect to varying parameters.

In addition to RINQ, we also report the running time of a hypothetical method named *Oracle*. We assume that Oracle already knows the networks in the result set for any query and aligns query with only those networks. Thus, the running time of Oracle is a lower bound to any database search method. The purpose we had for including such a hypothetical method is to observe how much room for improvement RINQ leaves in practice.

Effects of the number of references: Figure 4 shows the results for varying number of references. The results suggest that RINQ achieves up to 80% improvement in running time over exhaustive search. With higher indel penalties, the importance of matching topologies increase. Our experiments with RINQ show that as the alignment favors topology, the accuracy of our method increases. This suggests

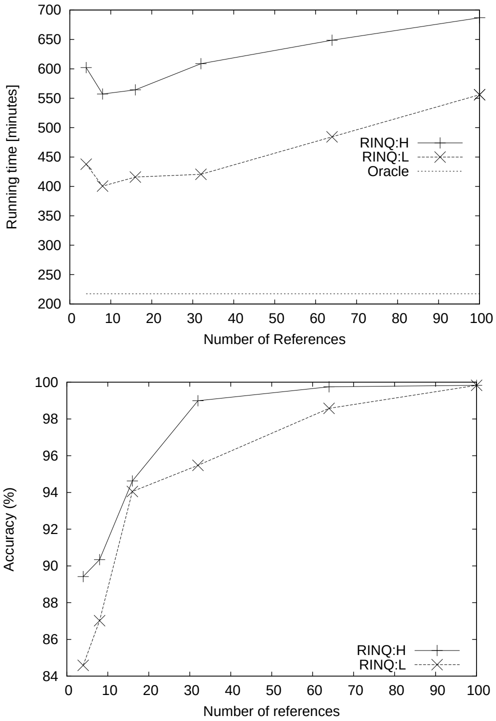


Fig. 4. Impact of the number of references. Top: The average running time RINQ and Oracle. Bottom: The accuracy of RINQ. The query selectivity is 8%. The graphs includes two different tests with RINQ. RINQ:H refers to RINQ experiments which use alignments with high indel penalties. RINQ:L refers to experiments using alignments with low indel penalties. The running time of exhaustive search is over two days (not shown in figure).

that the references can preserve the alternative topologies in the database accurately. Another observation from this experiment is, as the number of references increase, the running time of our method increases slowly while its accuracy increases rapidly to almost 100%. This is a desirable property as it suggests that RINQ needs to maintain only a small number of references. Thus, it uses a small amount of storage and performs a small number of query-reference network alignments. From our results, we suggest using around 30 references for our database.

Effects of query selectivity: Following from our previous experiments, we fix the number of references to 32 and vary the query selectivity. Figure 5 plots the results. As the selectivity increases, the size of the result set, and thus the number of networks we need to align with the query increases. As a result, the running time of RINQ grows. We observe that running time of our method does not increase linearly. This is expected as the size of the actual result set grows, it gets harder to place database networks into the filter set. The small gap between Oracle and RINQ indicates that our method is very successful in filtering unpromising database networks. It also shows that our method leaves little room for improvement particularly for small selectivity values. For instance, when the selectivity is 3 %, our method's running time is almost that of the smallest number of alignments needed for that selectivity.

For all the selectivity values, RINQ has very high accuracy. We observe that as we increase the query selectivity, RINQ's accuracy increases as well. This is mainly because for small selectivity values, even a few false dismissals can reduce the accuracy. The results

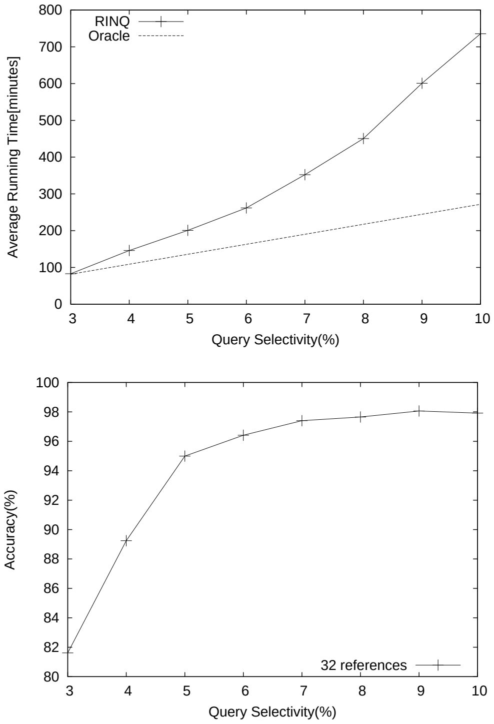


Fig. 5. Running time and accuracy of RINQ for different query selectivity values. These results are calculated using 32 references.

demonstrate that our method reaches to very high accuracy values quickly.

4.3 Comparison of RINQ with existing methods

An obvious question that needs to be answered is how well RINQ performs as compared to existing state of the art methods. In order to evaluate this, we compared RINQ with two existing methods, namely SAGA (Tian *et al.*, 2007) and CTree (He and Singh, 2006). In order to have a fair comparison, similar to RINQ, we used SAGA and CTree to create a candidate set of results. We used our QNet implementation to find the actual results from the candidate set for each method.

SAGA requires the user to provide the list of genes that can be aligned with each gene in the query network. The user can also specify the minimum percentage of query nodes that needs to be aligned. We varied these two parameters to get the smallest candidate set for SAGA for different accuracies. CTree uses a neighbor biased matching algorithm to filter database networks. We varied the constant that defines the amount of neighbor bias to get the best running time of CTree for different accuracy values.

We ran RINQ, SAGA and CTree using the same query and database networks. We queried each method with the same set of query selectivity values. We recorded running time and accuracy of each method. For all the three methods, the amount of time it takes for filtering the database was much smaller than the post-processing time, where we compare query network with the candidate networks. In our experiments, RINQ requires only eight seconds for filtering phase using 32 references, whereas, the other methods require from a few seconds up to minutes. We ignore the amount of time the index structure takes to filter database networks for this is a negligible fraction (0.03 % to 0.1 %) of the time it takes to process the candidate

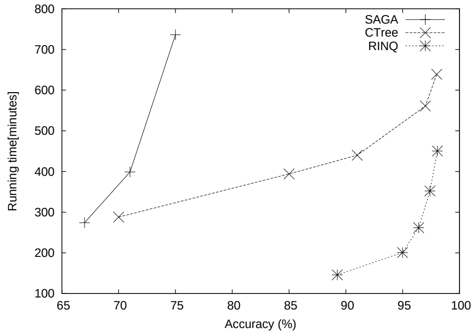


Fig. 6. Running time versus accuracy of SAGA, CTree and RINQ. Experiments are repeated for same queries with different selectivities. Average query processing time is presented as running time. Accuracy is calculated over all test queries. Lower running time and higher accuracy indicates better performance.

set. We report the average running time and accuracy values for SAGA, CTree and RINQ over all queries.

Figure 6 plots our results. In our experiments RINQ performed much better in terms of both accuracy and speed than both CTree and SAGA. RINQ is up to 3 times faster than CTree for the same accuracy values. We could not achieve comparable accuracy with SAGA. We believe that this is probably because SAGA is well suited for exact matches and its accuracy drops quickly as the difference between the genes in the query and the database network grows. In order to improve the accuracy of SAGA further, we grew the set of genes that each gene can align (i.e., this corresponds to reducing the similarity cutoff for gene pairs in the SAGA queries). However, SAGA fails to return any results in that case. Given that (1) its running time (when its accuracy is 75 %) is already much larger than both RINQ and CTree (when their accuracies are 85 and 89 % respectively), and (2) its running time will only increase with increased accuracy, we concluded that SAGA could not compete with the two for high accuracy values. *In conclusion, we observed that RINQ is a significant improvement over existing methods.*

4.4 Significance of results

In order to validate our method, we discuss the significance of our results in this section. We first evaluate the statistical significance of our results. We then discuss some of the biologically interesting observations from the statistically significant results.

We start by discussing the statistical significance of our results. We compute the statistical significance of each query as its p-value. We do this as follows for a given query network and a similarity cutoff. We first run RINQ to find the set of database networks that have an alignment with score greater than the cutoff with the query. We then compute the probability that a given random query network has at least as many matching database networks in the same dataset with alignment score at least the same similarity cutoff. We report this probability as the p-value.

Table 2 presents the p-values and the result set sizes of the top five queries when similarity cutoff is set to 90, 80 and 70 % of the maximum possible similarity score. The maximum score is the score that is achieved when the query network is aligned with itself. We observe that for all the three similarity cutoff values the top queries are statistically significant. This implies that the database contains motifs that have at least as many genes as the query networks. The top queries are a part of these motifs. *The results are also encouraging as they demonstrate that RINQ is capable of finding such frequent patterns quickly.*

Table 2. The p-values and the result set sizes of top five queries when similarity cutoff is set to 90, 80 and 70 % of the maximum similarity score that can be achieved with an exact match. The values in the parantheses are the result set size (i.e., the number of matches) reported by RINQ for that specific query.

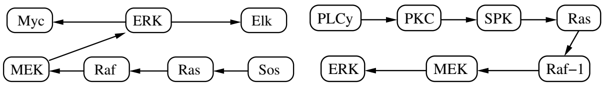
	Similarity [%]					
Rank	90		80		70	
1	1.4E-6	(21)	1.0E-13	(34)	<1E-15	(48)
2	5.7E-4	(17)	1.3E-7	(27)	<1E-15	(44)
3	3.4E-2	(13)	5.4E-4	(21)	1.9E-7	(32)
4	3.4E-2	(13)	4.4E-2	(16)	3.0E-6	(30)
5	7.3E-2	(12)	4.4E-2	(16)	3.2E-4	(26)

When the similarity cutoff decreases, the number of matching database networks grow. This proves that motifs are not exact duplicates of the same networks, but they are approximate alignments. This is an important result as it shows the necessity of approximate alignment algorithms. Decreasing p-values also indicate that our results are more significant. This supports our results in Figure 5(b) that for lower similarity cutoffs (i.e., higher query selectivity), RINQ has higher accuracy. Higher accuracy and lower p-values indicate that RINQ performs better with lower similarity cutoffs.

One can argue that the large number of matches found by RINQ can be attributed to the fact that there are multiple pathways that belong to the same pathway class from different organisms (See Table 4 in Appendix II). Next we focused on the individual queries that have the best p-value and explored the characteristics of the matching database subnetworks as well as the biological significance of these matches.

Figure 7 depicts two of our top queries. The query in Figure 7(a) is a subnetwork of the ErbB signaling network of *Rattus norvegicus*. This query is ranked as the first for 90 and 80 % similarity and the second for the 70 % similarity. ErbB signaling network is closely related to cancer (Hynes and Lane, 2005). Mutations and dysregulations of ErbB network proteins are frequently observed in cancer cells. ErbB signaling network consists of ErbB family of receptor proteins and a number of intracellular signaling pathways (Yarden and Sliwkowski, 2001). ErbB family of receptors are cell membrane proteins. Extracellular growth factors bind to these proteins. In response to this binding, ErbB proteins can trigger many different cell signaling networks. One of the most important intracellular targets of ErbB receptors is MAPK signaling network. Our first query is a part of ErbB signaling pathway that is responsible for triggering the MAPK signaling pathway.

RINQ successfully identifies the relationship between our query and the similar patterns in the ErbB and MAPK signaling networks of other organisms. When we query our database for 90% similarity, RINQ successfully returns all the 21 true results, with only three false positives. Among these 21 results, nine are from the ErbB signaling network of various organism and the rest from the MAPK signaling network. For queries with 80% and 70% similarity, RINQ also identifies the database networks with 100% accuracy. For 80 % similarity, RINQ finds matching patterns in the Insulin signaling networks of nine organisms. Indeed, the insulin signaling network contains the Ras-Raf-MEK-ERK-Elk1 pathway that is responsible from growth and differentiation and the Ras-Raf-MEK-ERK-MNK pathway that contributes to protein synthesis. As we reduce the similarity to 70 %, RINQ returns additional alignments from the



(a) (b)

Fig. 7. Sample queries from our experiments. (a) A subnetwork of ErbB signaling pathway of *Rattus Norvegicus* (rat). (b) A subnetwork of VEGF signaling pathway of *Mus Musculus* (mouse)

chemokine signaling pathway. There are totally five organisms, namely *Mus musculus*, *Homo sapiens*, *Canis familiaris*, *Bos taurus* and *Pan troglodytes*, whose chemokine signaling pathways align with this query. These are all phylogenetically close to the query organism. Further exploration of these pathways show that they indeed contain the Sos-Ras-Raf-MEK1-ERK1/2 pathway that regulates a set of critical functions such as cytokine production, cellular growth and differentiation, cell survival, apoptosis and migration. Some organisms, such as *Ornithorhynchus anatinus*, *Sus scrofa*, *Gallus gallus* and *Xenopus laevis* have one or more protein deletions in their ErbB signaling networks. Among these the first two belong to the same class as the query organism (i.e., mammalian) while the last two belong to different classes. Nevertheless, RINQ successfully identifies such networks and provides tight estimates for their actual network similarity.

The query in Figure 7(b) is taken from the VEGF signaling pathway. Although there are VEGF signaling networks of only 14 organisms in our database, this query returns totally 48 alignments at 70 % similarity. These alignments are found in eight types of networks of 14 different organisms. This demonstrates that the VEGF signaling network interacts with many other networks. It also suggests that this query is a part of an important motif as it appears in numerous organisms. RINQ could identify this motif as well among others.

5 CONCLUSION AND FUTURE WORK

In this paper, we considered similarity queries in biological network databases. Since pairwise comparison of biological networks is a time consuming operation, exhaustive database searches are impractical for such databases. We introduced reference based indexing to speed up such queries. We developed methods for calculating upper and lower bounds to the alignment score quickly. Using these bounds, we classified database networks as similar or not without aligning them with the query network. We also developed an intelligent method to choose references which maximizes the chance of references to prune database networks. Our experiments showed that (1) our method reduced the running time of a single query on a database of around 300 networks from over two days to only eight hours; (2) our method outperformed the state of the art method Closure Tree and SAGA by a factor of three or more; (3) our method successfully identified statistically and biologically significant relationships across networks and organisms. To the best of our knowledge this is the first attempt at developing reference based indexing of biological networks. Therefore, application of this method to other types of biological networks is an important future research direction for us. This approach enables indexing large biological networks. Therefore, we believe that it is an important step towards large scale analysis of biological network databases and has great potential to assist biological sciences which need such large scale comparison.

Acknowledgments. This work was supported partially by NSF under grants CCF-0829867 and IIS-0845439.

REFERENCES

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403 – 410.

Ay, F. and Kahveci, T. (2010). SubMAP: Aligning Metabolic Pathways with Subnetwork Mappings. In *RECOMB*, pages 15–30.

Ay, F., Kahveci, T., and de Crécy-Lagard, V. (2009). A fast and accurate algorithm for comparative analysis of metabolic pathways. *J. Bioinformatics and Computational Biology*, **7**(3), 389–428.

Banks, E., Nabieva, E., Peterson, R., and Singh, M. (2008). Netgrep: fast network schema searches in interactomes. *Genome Biology*, **9**(9), R138.

Bruckner, S., Hüffner, F., Karp, R. M., Shamir, R., and Sharan, R. (2009). Torque: topology-free querying of protein interaction networks. *Nucleic Acids Research*, **37**(suppl 2), W106–W108.

Clemente, J. C., Satou, K., and Valiente, G. (2005). Reconstruction of phylogenetic relationships from metabolic pathways based on the enzyme hierarchy and the gene ontology. *Genome Inform*, **16**(2), 45–55.

Clemente, J. C., Satou, K., and Valiente, G. (2006). Finding conserved and non-conserved reactions using a metabolic pathway alignment algorithm. *Genome Inform*, **17**(2), 46–56.

Dost, B., Shlomi, T., Gupta, N., Ruppin, E., Bafna, V., and Sharan, R. (2008). QNet: a tool for querying protein interaction networks. *J Comput Biol*, **15**(7), 913–925.

Ferro, A., Giugno, R., Pigola, G., Pulvirenti, A., Skripin, D., Bader, G. D., and Shasha, D. (2007). NetMatch: a Cytoscape plugin for searching biological networks. *Bioinformatics*, **23**(7), 910–912.

Francke, C., Siezen, R. J., and Teusink, B. (2005). Reconstructing the metabolic network of a bacterium from its genome. *Trends Microbiol*, **13**(11), 550–558.

Giot, L. and Bader et al, J. S. (2003). A protein interaction map of drosophila melanogaster. *Science*, **302**(5651), 1727–1736.

Giugno, R. and Shasha, D. (2002). GraphGrep: A fast and universal method for querying graphs. In *Proc. 16th Int Pattern Recognition Conf*, volume 2, pages 112–115.

He, H. and Singh, A. K. (2006). Closure-Tree: An index structure for graph queries. *International Conference on Data Engineering*, **0**, 38.

Hynes, N. E. and Lane, H. A. (2005). Erbb receptors and cancer: the complexity of targeted inhibitors. *Nat Rev Cancer*, **5**(5), 341–354.

Kelley, B. P., Yuan, B., Lewitter, F., Sharan, R., Stockwell, B. R., and Ideker, T. (2004). PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Res*, **32**(Web Server issue), W83–W88.

Levine, M. and Davidson, E. H. (2005). Gene regulatory networks for development. *Proceedings of the National Academy of Sciences of the United States of America*, **102**(14), 4936–4942.

Liao, C., Lu, K., Baym, M., Singh, R., and Berger, B. (2009). IsoRankN: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, **25**(12), 253–238.

Mongiovi, M., Natale, R. D., Giugno, R., Pulvirenti, A., Ferro, A., and Sharan, R. (2010). SIGMA: a set-cover-based inexact graph matching algorithm. *J. Bioinformatics and Computational Biology*, **8**(2), 199–218.

Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., and Kanehisa, M. (1999). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res*, **27**(1), 29–34.

Pinter, R. Y., Rokhlenko, O., Yeger-Lotem, E., and Ziv-Ukelson, M. (2005). Alignment of metabolic pathways. *Bioinformatics*, **21**(16), 3401–3408.

Shlomi, T., Segal, D., Ruppin, E., and Sharan, R. (2006). QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, **7**, 199.

Singh, R., Xu, J., and Berger, B. (2007). Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology. In *RECOMB*, pages 16–31.

Sridhar, P., Kahveci, T., and Ranka, S. (2007). An iterative algorithm for metabolic network-based drug target identification. *Pacific Symposium on Biocomputing*, pages 88–99.

Tian, Y., McEachin, R. C., Santos, C., States, D. J., and Patel, J. M. (2007). SAGA: a subgraph matching tool for biological graphs. *Bioinformatics*, **23**(2), 232–239.

Yan, X., Yu, P. S., and Han, J. (2004). Graph indexing: a frequent structure-based approach. In *SIGMOD*, pages 335–346, New York, NY, USA. ACM.

Yarden, Y. and Sliwkowski, M. X. (2001). Untangling the erbb signalling network. *Nat Rev Mol Cell Biol*, **2**(2), 127–137.

APPENDIX I. TIME COMPLEXITY OF THE METHODS DEVELOPED IN THIS PAPER

In this section, we present the time complexity of the methods we developed throughout this paper. We will assume that all the necessary network alignments are precomputed and stored such that we can retrieve them in linear time. For simplicity, we represent the number of nodes networks q and d with $|q|$ and $|d|$ respectively.

Complexity of lower bound calculation

We calculate the complexity of lower bound calculation for each phase separately. In Phase I, we go over all the nodes in q, to find the composite relation δ. So, the time complexity of Phase I is $O(|q|)$. In Phase II, we traverse all nodes in q to complete the alignment. The complexity of this phase is $O(|q|)$. Finally, we calculate the scores by adding all the individual scores accumulated by each node in q, which is $O(|q|)$. So, the overall time complexity of lower bound calculation is $O(|q|)$.

Complexity of approximate upper bound

Complexity of upper bound computation can be calculated in two phases. In the first phase, the algorithm finds the indirect mappings through each of the k references. The complexity of this phase is $O(k|q|)$. In the second phase, we perform MWBM between the nodes of q and d. Using Hungarian algorithm, we can execute this matching in $O(|d|^2|q|)$ time. In total, the time complexity of calculating $UB(q, d)$ is $O(k|q| + |d|^2|q|)$.

Complexity of candidate reference creation

To create the candidate reference set, we iterate the following two steps:

Step I. Create a seed using random walk

Step II. Align current seed with the rest of the candidate reference networks

We analyze each of the two steps next.

Step I. We perform random walk simply by listing our possible choices of nodes and then choosing a new node randomly from this list. We choose T nodes in order to create our seed. Updating the set of possible choices after each node selection takes $O(|d|)$ time. Thus, the total time to generate a seed with T nodes is $O(T|d|)$.

Step II. After creating a seed, we align this seed with all the existing m candidate references in C. Recall that the number of nodes in each network in C is T. In other words, the number of nodes in the seed and the candidate networks are the same. This property of the two networks enable us to align them optimally using dynamic programming in a single iteration. Thus, the complexity of a single seed to candidate reference alignment is $O(2^{2^T}T)$. There are at most m networks in C. Hence, m alignments take $O(m2^{2^T}T)$.

From the complexities derived above, we conclude that a single iteration takes $O(m2^{2^T}T + T|d|)$ time. We repeat this until the iterations fail a predefined number of times consecutively. Let's denote this number with f. In the worst case, we have to try $f - 1$ times to create the next reference. To create m reference networks, we need to try $1 + (m - 2)(f - 1) + f$ times, which is $O(mf)$. So, the overall complexity of candidate reference creation can be expressed as $O(mf(m2^{2^T}T + T|d|))$. In our experiments, we were able to create C in approximately 10 minutes, where $m = 100$, $f = 100$, and $T = 7$.

Complexity of reference selection

We consider the complexity of Algorithm 1 in two separate parts.

Part I. (PREPROCESSING) First, we align all (q, d), (q, c) and (c, d) pairs. Let t denote the number of training queries. The complexity of the alignment method we use defines the time complexity of these $tn + tm + mn$ alignments. We are using QNet algorithm to align networks. so, the complexity of the preprocessing phase is $O(tn(2^{|q||d|}|d|)\rho_1 + tm(2^{|q||c|}|c|)\rho_2 + mn(2^{|c||d|}|d|)\rho_3)$ where ρ_i is the number of iterations per alignment. Number of iterations depends on the number of nodes in the smaller network and the predefined confidence value for the result.

Part II. (POSTPROCESSING) Once we compute all the pairwise alignments described above, we are ready to select references. Our algorithm iterates until it finds the reference set with largest accuracy value. During these iterations, first we calculate the accuracy of the index for using each $R - \{r_i\}$ as the reference set. To do this, we compute the upper bound for each (q, d) pair. In total, there are tnk such upper bound calculations, resulting in $O(tnk(k|q| + |d|^2|q|))$ time. We also compute the accuracy using each $R \cup \{c_i\}$. This step takes $O(tmk(k|q| + |d|^2|q|))$ time. In the worst scenario, this iteration repeats until all the networks in C are used. So, the overall complexity of the postprocessing is $O(((tnk(k|q| + |d|^2|q|)) + (tmk(k|q| + |d|^2|q|))) * m)$.

In our experiments, we used QNet algorithm with 99% confidence to align two biological networks. Using this alignment method, we aligned 50 training queries, 100 candidate references and 297 database networks with each other. These alignments took around 25 days for seven node query and reference networks on a single cpu. However, the iterations of the algorithm completed within a few hours after completing the alignments.

APPENDIX II. DETAILS OF THE SAMPLE DATASET

We downloaded our sample database from KEGG (Ogata *et al.*, 1999). We downloaded all the gene regulatory networks that are larger than 15 nodes. In total, our database consists of 297 gene regulatory networks. In this database, we had 21 different type of networks from 46 different organisms. Table 3 shows the different organisms and the number of networks which belong to that organism.

Homo sapiens, *Mus musculus* and *Rattus norvegicus* have the largest number of networks among all the organisms in our database.

Table 4 displays the different types of networks in our database. The most frequent pathway is the Phosphatidylinositol signaling system which appears in 45 different organisms. Note that the same network of different organisms can differ in terms of the genes as well as the interactions that govern them.

Figure 8 presents the distribution of the number of genes and the number of interactions of each network in our database. Half of the networks in our database has 30 or more genes. Also, half of the networks in our database has more than 40 interactions. About 6 % of these have more than 100 nodes and interactions.

Figure 9 plots the density of the networks in the database in terms of the number of interactions as compared to the number of genes. The number of interactions are usually slightly more than the number of genes, indicating that the networks are often sparse.

Table 4. The distribution of the networks in our database to different classes of networks. Each network class corresponds to a different biological function. There are totally 21 classes of networks in our database. The networks are listed in alphabetical order.

Network name	Num of networks
Adipocytokine signaling pathway	16
B cell receptor signaling pathway	9
Calcium signaling pathway	11
Chemokine signaling pathway	11
ErbB signaling pathway	18
Fc epsilon RI signaling pathway	10
Fc gamma R-mediated phagocytosis	9
GnRH signaling pathway	3
Insulin signaling pathway	14
Jak-STAT signaling pathway	15
MAPK signaling pathway	18
MAPK signaling pathway - yeast	1
mTOR signaling pathway	10
NOD-like receptor signaling pathway	12
Phosphatidylinositol signaling system	45
PPAR signaling pathway	16
RIG-I-like receptor signaling pathway	5
T cell receptor signaling pathway	10
Toll-like receptor signaling pathway	14
VEGF signaling pathway	14
Wnt signaling pathway	36

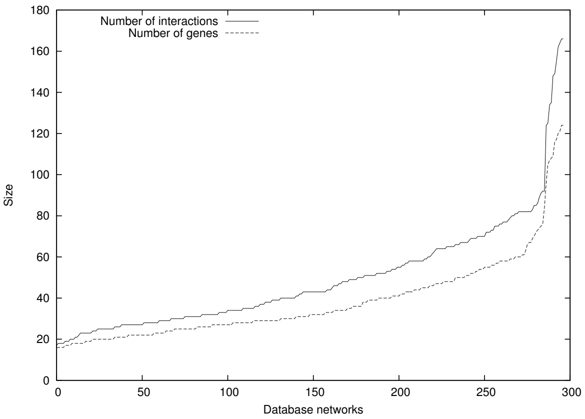


Fig. 8. The distributions number of genes and the number of interactions for each network in our database. The graphs does not show a relation between the number of genes and number of interactions.

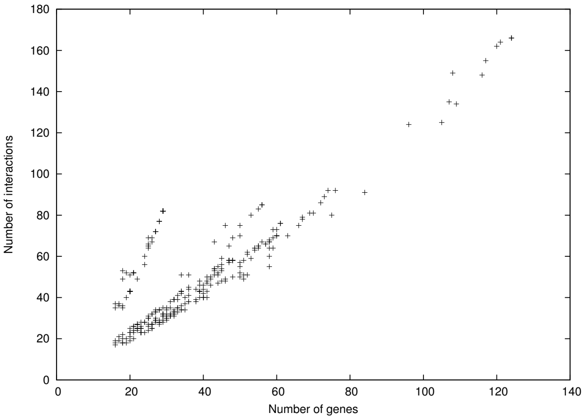


Fig. 9. The number of genes and number of interactions for each network in our database.

Table 3. The distribution of the networks in our database to different organisms. There are totally 46 organisms in our database. The organisms are listed in alphabetical order.

Organism	Num of networks	Organism	Num of networks
Aedes aegypti (yellow fever mosquito)	2	Drosophila simulans	1
Anopheles gambiae (mosquito)	2	Drosophila yakuba	2
Apis mellifera (honey bee)	2	Equus caballus (horse)	19
Acyrthosiphon pisum (pea aphid)	2	Gallus gallus (chicken)	13
Arabidopsis thaliana (thale cress)	1	Hydra magnipapillata	2
Branchiostoma floridae (Florida lancelet)	1	Homo sapiens (human)	20
Brucella melitensis bv. 1 16M	4	Monosiga brevicollis	1
Bos taurus (cow)	19	Macaca mulatta (rhesus monkey)	17
Caenorhabditis briggsae	2	Monodelphis domestica (opossum)	17
Caenorhabditis elegans (nematode)	3	Mus musculus (mouse)	20
Canis familiaris (dog)	18	Nematostella vectensis (sea anemone)	2
Ciona intestinalis (sea squirt)	2	Nasonia vitripennis (jewel wasp)	2
Culex quinquefasciatus (southern house mosquito)	2	Ornithorhynchus anatinus (platypus)	13
Drosophila ananassae	2	Pan troglodytes (chimpanzee)	16
Dictyostelium discoideum (cellular slime mold)	1	Rattus norvegicus (rat)	20
Drosophila erecta	2	Saccharomyces cerevisiae (budding yeast)	1
Drosophila grimshawi	2	Strongylocentrotus purpuratus (purple sea urchin)	2
Drosophila melanogaster (fruit fly)	3	Sus scrofa (pig)	8
Drosophila mojavensis	2	Trichoplax adhaerens	1
Drosophila persimilis	2	Tribolium castaneum (red flour beetle)	2
Drosophila pseudoobscura pseudoobscura	2	Taeniopygia guttata (zebra finch)	13
Danio rerio (zebrafish)	13	Xenopus laevis (African clawed frog)	8
Drosophila sechellia	2	Xenopus tropicalis (western clawed frog)	6