

# QNet: A Tool for Querying Protein Interaction Networks

Banu Dost<sup>1,\*</sup>, Tomer Shlomi<sup>2,\*</sup>, Nitin Gupta<sup>1</sup>, Eytan Ruppin<sup>2,3</sup>,  
Vineet Bafna<sup>1</sup>, and Roded Sharan<sup>2</sup>

<sup>1</sup> Computer Science and Engineering,  
Univ. of California, San Diego, CA 92093, USA

{bdost,ngupta,vbafna}@cs.ucsd.edu

<sup>2</sup> School of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel  
{shlomito,ruppin,roded}@post.tau.ac.il

<sup>3</sup> School of Medicine, Tel Aviv University, 69978 Tel Aviv, Israel

**Abstract.** Molecular interaction databases can be used to study the evolution of molecular pathways across species. Querying such pathways is a challenging computational problem, and recent efforts have been limited to simple queries (paths), or simple networks (forests). In this paper, we significantly extend the class of pathways that can be efficiently queried to the case of trees, and graphs of bounded treewidth. Our algorithm allows the identification of non-exact (homeomorphic) matches, exploiting the color coding technique of Alon et al. We implement a tool for tree queries, called QNet, and test its retrieval properties in simulations and on real network data. We show that QNet searches queries with up to 9 proteins in seconds on current networks, and outperforms sequence-based searches. We also use QNet to perform the first large scale cross-species comparison of protein complexes, by querying known yeast complexes against a fly protein interaction network. This comparison points to strong conservation between the two species, and underscores the importance of our tool in mining protein interaction networks.

## 1 Introduction

The study of biological networks has gained substantial interest in recent years. In particular, technological advances, such as the yeast two-hybrid [11] and co-immunoprecipitation assays [15], have enabled the large-scale mapping of protein-protein interactions (PPIs) across many model species. The newly available PPI networks present a host of new challenges in studying protein function and evolution. Key to addressing these challenges is the development of efficient tools for network database searches, much the same as sequence searches have been instrumental in addressing similar problems at the genome level.

Network queries call for searching a “template” subnetwork within a network of interest. Commonly, the query is a known pathway, and the network is searched for subnetworks that are similar to the query. Similarity is measured

---

\* These authors contributed equally to this work.

both in terms of protein sequence similarity and in terms of topological similarity. The hardness of the problem stems from the non-linearity of a network, making it difficult to apply sequence alignment techniques for its solution.

Several authors have studied the network querying problem, mostly focusing on queries with restricted topology. Kelley et al. [13] devised an algorithm for querying linear pathways in PPI networks. While the problem remains NP-hard in this case as well (as, e.g., finding the longest path in a graph is NP-complete [7]), an efficient algorithm that is polynomial in the size of the network and exponential in the length of the query was devised for it. Pinter et al. [17] enable fast queries of more general pathways that take the form of a tree. However, their algorithm is limited to searching within a collection of trees rather than within a general network. Sohler and Zimmer [6] developed a general framework for subnetwork querying, which is based on translating the problem to that of finding a clique in an appropriately defined graph. Due to its complexity, their method is applicable only to very small queries. Recently, some of us have provided a comprehensive framework, called QPath, for linear pathway querying. QPath is based on an efficient graph theoretic technique, called color coding [1], for identifying subnetworks of “simple” topology in a network. It improves upon [13] both in speed and in higher flexibility in non-exact matches.

In this paper, we greatly extend the QPath algorithm to allow queries with more general structure than simple paths. We provide an algorithmic framework for handling tree queries under non-exact (homeomorphic) matches (Section 3.1). In this regard, our work extends [17] to querying within general networks, and the results in [1] to searching for homeomorphic rather than isomorphic matches. More generally, we provide an algorithm for querying subnetworks of bounded treewidth (Section 3.2). We implemented a tool for tree queries which we call QNet. We demonstrate that QNet performs well both in simulation of synthetic pathway queries, and when applied to mining real biological pathways (Section 5). In simulations, we show that QNet can handle queries of up to 9 proteins in seconds in a network with about 5,000 vertices and 15,000 interactions, and that it outperforms sequence-based searches. More importantly, we use QNet to perform the first large scale cross-species comparison of protein complexes, by querying known yeast complexes in the fly protein interaction network. This comparison points to strong conservation of protein complexes structures between the two species. For lack of space some algorithmic details are omitted in the sequel.

## 2 The Graph Query Problem

Let  $G = (V, E, w)$  be an undirected weighted graph, representing a PPI network, with a vertex set  $V$  of size  $n$ , representing proteins, an edge set  $E$  of size  $m$ , representing interactions, and a weight function  $w : E \rightarrow \mathcal{R}$ , representing interaction reliabilities.

Let  $G_Q = (V_Q, E_Q)$  denote a query graph with  $k$  vertices. We reserve the term *node* for vertices of  $G_Q$  and use the term *vertex* for vertices of  $G$ .

Let  $h(q, v)$  denote a similarity score between query node  $q \in V_Q$  and vertex  $v \in V$ . In our context, vertices correspond to proteins, and their similarity score is a function of their sequence similarity. A query node  $q$  is referred to as *homologous* to a graph vertex  $v$ , if the corresponding similarity score  $h(q, v)$  exceeds a predefined threshold.

A *subdivision* of an edge  $(u, v)$  in a graph  $H = (U, F)$  replaces it with two edges  $(u, w)$  and  $(w, v)$ , where  $w \notin U$ , i.e., creating a new graph  $H' = (U \cup \{w\}, F \cup \{(u, w), (w, v)\} \setminus \{(u, v)\})$ .  $H$  is considered *extendable* to a graph  $G$ , if  $G$  can be obtained from  $H$  by a series of subdivisions. In particular,  $H$  is then homeomorphic to  $G$ .

An *alignment* of the query graph  $G_Q$  to  $G$  is defined as a pair of: (i) a subgraph  $G_A = (V_A, E_A)$  of  $G$ , referred to as the *alignment subgraph*; and (ii) a bijection,  $\sigma : V_Q^S \rightarrow V_A^S$ , between a subset of query nodes,  $V_Q^S \subseteq V_Q$ , and homologous vertices in the alignment subgraph,  $V_A^S \subseteq V_A$ . The vertices in  $V_Q^S \cup V_A^S$  are called *skeleton* vertices. Pairs of associated vertices  $(q, \sigma(q)) \in V_Q^S \times V_A^S$  are called *aligned*.

An alignment is *proper* if there exists a pair of *skeleton* graphs  $S_Q = (V_Q^S, E_Q^S)$  and  $S_A = (V_A^S, E_A^S)$  that satisfy the following conditions: (i) there is an isomorphism between  $S_Q$  and  $S_A$  which respects the alignment (i.e., there is an edge  $(u, v) \in E_Q^S$  iff there is an edge  $(\sigma(u), \sigma(v)) \in E_A^S$ ); and (ii)  $S_Q$  is extendable to  $G_Q$  and  $S_A$  is extendable to  $G_A$ . In particular, this means that  $G_Q$  and  $G_A$  are required to be homeomorphic. In the rest of the paper we discuss proper alignments only. An example of such an alignment is given in Figure 1a.

Query nodes that are not aligned with vertices in the alignment subgraph are considered to be *deleted*. Conversely, vertices in the alignment subgraph that are not aligned with query nodes are considered to be *inserted*. Insertions and deletions are also referred to as *indels*. From the above definitions, inserted and deleted vertices must be of degree 2 in their respective graphs. An alignment which involves no insertions or deletions is considered *simple*. The weight of an alignment is the sum of: (i) similarity scores of aligned vertices, (ii) weights of edges in the aligned subgraph, (iii) a penalty score,  $\delta_d$ , for each node deletion, and (iv) a penalty score,  $\delta_i$ , for each vertex insertion.

The *graph query problem* is formally defined as follows: Given a query graph  $G_Q$ , a graph  $G$ , a similarity score  $h$ , and penalty scores for insertions and deletions, find a proper alignment of  $G_Q$  in  $G$  with maximal weight. In practice, we would also like to limit the number of insertions and deletions in the alignment, to control the evolutionary distance between the two subnetworks. To this end, we also consider a variant of the problem in which the number of insertions is limited by  $N_{ins}$ , and the number of deletions is limited by  $N_{del}$ .

### 3 Graph Query Algorithms

The complexity of the graph query problem depends on the topology of the query graph  $G_Q$ , the topology of the graph  $G$ , and the similarity function  $h$ . In the general case, the problem of finding simple alignments is in general equivalent to

subgraph isomorphism [8], which is computationally hard. In this paper, we focus on efficient query algorithms by exploiting the underlying biological constraints. Specifically, motivated by known pathways in KEGG [12], we consider restricted query topologies, i.e., the query graph being a *tree*, and a graph of *bounded treewidth* (see also [17]). For these special structures, we adapt the color coding method of Alon et al. [1] to make the problem tractable.

Color coding is a randomized technique for finding simple paths and simple cycles of a specified length  $k$  within a given graph of size  $n$ . The basic idea is to randomly assign  $k$  colors to the vertices of the graph and then search for *colorful* paths in which each color is used exactly once. Thus, rather than having to maintain a list of vertices visited so far (of size  $O(n^k)$ ), one can maintain a list of colors at considerably lower complexity ( $O(2^k)$ ).

The use of the color coding technique within a query algorithm is intuitively similar. We construct an optimal alignment by extending optimal sub-alignments using dynamic programming. Adding a network vertex to the optimal alignment can be done only if this vertex is not already contained in the sub-optimal alignment. Thus, naively, each potential sub-optimal alignment should maintain the list of at most  $k$  vertices already matched. This yields  $O(n^k)$  potential alignments. In color coding, we a priori color each network vertex randomly with one of  $k$  colors, looking for a colorful alignment. Consequently, we only need to maintain a list of used colors (of size  $O(2^k)$ ), which significantly reduces the computation time. However, the computation returns a correct answer only if the optimum alignment is colorful, which happens with probability  $\frac{k!}{k^k} \simeq e^{-k}$ . Therefore, if we repeat the experiment  $\ln(\frac{1}{\epsilon})e^k$  times, we get the optimum alignment with probability at least  $1 - \epsilon$  for any desired value of  $\epsilon$ .

### 3.1 Tree Query

We describe an algorithm for solving the graph query problem assuming that the query graph is a tree. For ease of presentation, we start by presenting a simplified version of the algorithm that limits the number of insertions only. The proper treatment of limiting both the number of insertions and deletions is deferred to the end of the section.

First, we root  $G_Q$  arbitrarily at a node  $r$  with degree 1. For each query node  $q$ , denote its children by  $q_1, \dots, q_{n_q}$ , where  $n_q$  denotes their number. Let  $T_{q,j}$  denote the tree that includes  $q$  and the subtrees rooted at each of its first  $j$  children, for  $1 \leq j \leq n_q$ . The algorithm proceeds in a series of trials in which every vertex  $v \in V$  is independently assigned a color  $c(v)$  drawn uniformly at random from the set  $C = \{1, 2, \dots, k + N_{ins}\}$ . Given the random vertex colors, we employ dynamic programming to identify an optimal colorful alignment. Let  $W^M(q, v, S, j)$  denote the maximal score of an alignment of  $T_{q,j}$  in  $G$ , such that query node  $q$  is aligned with graph vertex  $v$ , with the aligned subgraph receiving distinct colors from  $S \subseteq C$ . The recursion is initialized by setting  $W^M(q, v, S, 0) = h(q, v)$  for leaf nodes  $q$ , and is formulated as follows:

$$W^M(q, v, S, j) = \max_{\substack{u : (u, v) \in E \\ S' \subset S}} \begin{cases} (* \text{ Match, child } j *) \\ W^M(q, v, S', j-1) + W^M(q_j, u, S - S', n_{q_j}) + w(u, v), \\ \\ (* \text{ Insertion, vertex } u *) \\ W^M(q, v, S', j-1) + W^I(q_j, u, S - S') + w(u, v), \\ \\ (* \text{ Deletion, child } j *) \\ W^M(q, v, S', j-1) + W^D(q_j, v, S - S') \end{cases}$$

Here  $W^I(q, v, S)$  denotes the optimal score of an alignment of  $T_{q, n_q}$  in  $G$ , such that  $q$  is aligned with some vertex  $u$  that is a descendant of  $v$  in the aligned subgraph.  $W^D(q, v, S)$  denotes the optimal score of the alignment of  $T_{q, 1}$  in  $G$ , such that  $q$  is deleted and  $v$  is aligned with an ancestor of  $q$ . The recursions for the insertion and deletions cases are given below. For query nodes  $q$  of degree other than 2, we set  $W^D(q, v, S) = -\infty$ .

$$W^I(q, v, S) = \max_{u : (u, v) \in E} \begin{cases} W^M(q, u, S - \{c(v)\}, n_q) + w(u, v) + \delta_i, \\ W^I(q, u, S - \{c(v)\}) + w(u, v) + \delta_i \end{cases}$$

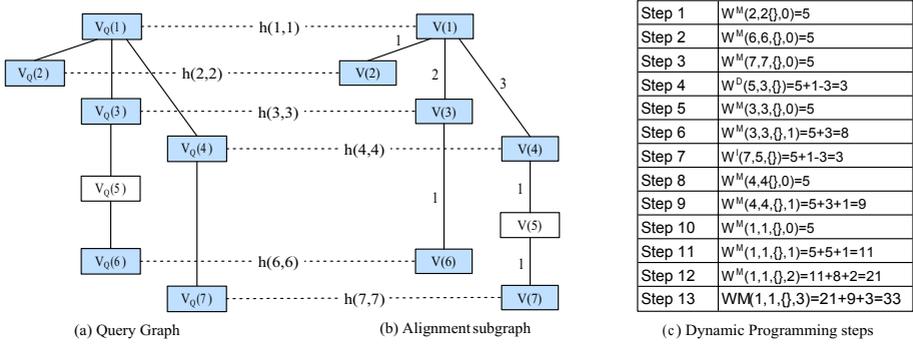
$$W^D(q, v, S) = \max_{u : (u, v) \in E} \begin{cases} W^M(q_1, u, S, n_{q_1}) + w(u, v) + \delta_d, \\ W^I(q_1, u, S) + w(u, v) + \delta_d, \\ W^D(q_1, v, S) + \delta_d \end{cases}$$

The maximal score of the alignment is  $\max_{v, S} W^M(r, v, S, 1)$ . The optimal alignment is obtained through standard dynamic programming backtracking. An application of the dynamic programming recursions to a sample query is demonstrated in Figure 1.

The running time of each trial is  $2^{O(k+N_{ins})}m$ . The probability of receiving distinct colors for the vertices of the optimal matching tree is at least  $e^{-k-N_{ins}}$ . Thus, the running time of the algorithm is  $2^{O(k+N_{ins})}m \ln(\frac{1}{\epsilon})$  for any desired success probability  $1 - \epsilon$  (where  $\epsilon > 0$ ). We note that it is straightforward to limit the number of deletions to  $N_{del}$  by incorporating an additional variable in the recursions to count the number of deletion in the optimal sub-alignment. The cost in terms of running time is multiplicative in  $N_{del}$ . When incorporating such a variable, it is also easy to limit the number of insertions to  $N_{ins}$  by choosing the optimum solution based on its number of deletions and the cardinality of its color set.

### 3.2 Bounded Treewidth Graph Query

The algorithm for matching trees can be extended to subgraphs that have tree-like properties. We present an algorithm for the simpler case where no indels are allowed and defer the description of an algorithm for the general case to the appendix. Intuitively, the treewidth of a graph indicates how close the graph is to being a tree, where a tree has treewidth 1. The maximal treewidth value for



**Fig. 1.** (a) An example of a tree query graph and the corresponding alignment subgraph. Numbers on the query graph’s edges represent an arbitrary ordering of children nodes. Aligned query nodes and graph vertices are connected with dashed lines. Nodes in the skeleton graphs appear in gray. (b) A simulation of the dynamic programming recursions. For simplicity, we denote color sets as  $\{\}$ . Matched vertices are awarded by  $+5$ , insertions and deletions are penalized by  $-3$  and edge weights are as shown.

a graph with  $n$  vertices is  $n - 1$  and this value is attained by an  $n$ -vertex clique. A formal definition of a treewidth and the associated tree-like structure follows.

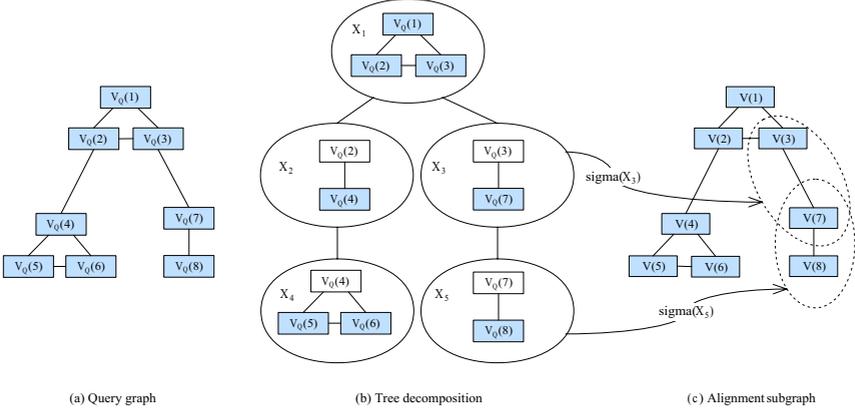
A *tree decomposition*  $(X, T)$  of the query graph  $G_Q = (V_Q, E_Q)$  is defined as follows (see, e.g., [14]):  $T = (I, F)$  is a rooted binary tree, and  $X = \{X_i \subseteq V_Q : i \in I\}$  is a collection of subsets of  $V_Q$ , such that  $\bigcup_{i \in I} X_i = V_Q$  and the following conditions are satisfied:

1. For each edge  $(u, v) \in E_Q$  there exists  $i \in I$  such that  $u, v \in X_i$ .
2. If  $i, j, k \in I$  and  $j$  is on the path from  $i$  to  $k$  in  $T$ , then  $X_i \cap X_k \subseteq X_j$ .

The *treewidth* of the tree decomposition is  $\max_{i \in I} |X_i| - 1$ . An example of a graph and its tree decomposition is given in Figure 2a,b.

Let  $t$  denote a bound on the treewidth of  $G_Q$ . We add a dummy node  $d$  as a parent of the root of  $T$ , with  $X_d = \emptyset$ . To avoid confusion, we call the nodes of  $T$ , *super-nodes*. For a non-leaf tree super-node  $X_i \in X$ , denote its two children by  $X_{i_1}$  and  $X_{i_2}$ . Let  $T_i$  denote the subtree of  $T$  that is rooted at  $X_i$ . The algorithm proceeds in a series of trials in which every vertex  $v \in V$  is independently assigned a color  $c(v)$  drawn uniformly at random from the set  $\{1, 2, \dots, k\}$ . Given the random vertex colors, we employ dynamic programming to identify an optimal colorful alignment.

The properties of the tree decomposition enable us to identify the optimal alignment by recursing on  $T$  and maintaining sub-optimal alignments of query nodes spanned by subtrees of  $T$ , similar to the tree query algorithm described above. However, there are two main difficulties to tackle: (i) A set of query nodes,  $X_i$ , may have an arbitrary topology (e.g., forming a clique), potentially requiring an exhaustive  $O(n^{t+1})$ -time search of an alignment subgraph for it. (ii) A query node  $v$  may appear in more than a single super-node.



**Fig. 2.** (a) An example of a query graph with a treewidth of 2. (b) A tree decomposition of the query graph such that each super-node has no more than 3 query nodes associated with it. Non-active query nodes are grayed. (c) An alignment subgraph.  $\sigma(X_3)$  and  $\sigma(X_5)$  are mappings of the query nodes in  $X_3$  and  $X_5$  to graph vertices, respectively, that identify on the active query node  $V_Q(6)$  in  $X_5$ .

For the first issue, we exploit the fact that the treewidth is bounded by  $t$ . Large values of  $t$  would make the algorithm impractical. To cope with the second difficulty, we note that by definition, if  $v \in X_{i_j}$  and  $v \notin X_i$ , then  $v \notin X_l$  for all super-nodes  $X_l$  that are not descendants of  $X_i$  in the tree. Thus, when visiting a certain super-node  $X_{i_j}$ , it contains *active* query nodes  $X_{i_j}^A = X_i \cap X_{i_j}$  that are yet to be handled, and *non-active* nodes  $X_{i_j}^N$  that can be removed from consideration when traversing up the tree (Figure 2b). We define a *non-active* edge at a super-node  $X_i$ , as a query edge touching a non-active node in  $X_i$ . We let  $E_i^N$  denote the set of non-active edges in super-node  $X_i$ .

We need some more notation before giving the main recurrence of the algorithm. For each  $X_i \in X$ , let  $\Sigma_i$  denote the  $O(n^{t+1})$ -size set of all mappings  $\sigma : X_i \rightarrow V$  such that: (i) for all distinct  $q_1, q_2 \in X_i$ ,  $c(\sigma(q_1)) \neq c(\sigma(q_2))$ ; and (ii) if  $(q_1, q_2) \in E_Q$  then  $(\sigma(q_1), \sigma(q_2)) \in E$ . Figure 2b,c shows an example of mappings between query nodes and graph vertices.

For computing the weight of an alignment, it is convenient to credit each super-node  $i$  (when traversing up the tree) with the similarity scores associated with its non-active nodes and the edge weights corresponding to its non-active edges. The node term is  $W^S(i, \sigma) = \sum_{u \in X_i^N} h(u, \sigma(u))$ . The edge term is  $W^E(i, \sigma) = \sum_{(u_1, u_2) \in E_i^N} w(\sigma(u_1), \sigma(u_2))$ .

Let  $W(i, \sigma, S)$  be the maximum weight of an alignment of a subgraph of  $G_Q$  that includes all super-nodes in  $T_i - X_i$ , identifies on the active query nodes in super-node  $i$  with the assignment  $\sigma \in \Sigma_i$ , and uses the colors in  $S \subseteq C$ .  $W(i, \sigma, S)$  can be recursively computed as follows. For a leaf  $i$ ,  $W(i, \sigma, S) = 0$ . For all other super-nodes:

$$W(i, \sigma, S) = \max_{S_1 \uplus S_2 = S} \sum_{j=1}^2 [W(i_j, \sigma_j, S_j) + W^S(i_j, \sigma_j) + W^E(i_j, \sigma_j)]$$

$$\sigma_1, \sigma_2$$

where  $\sigma$  is consistent with  $\sigma_1 \in \Sigma_{i_1}$  and  $\sigma_2 \in \Sigma_{i_2}$ .

The score of an optimal alignment of  $G_Q$  is thus  $\max_S W(d, \emptyset, S)$ . The total running time is  $2^{O(k)} n^{t+1}$ .

## 4 Implementation Notes

We implemented a tool, QNet, for querying a given network with a tree subnetwork, following the algorithm given in Section 3.1. Bounded treewidth queries will be supported in future versions. To allow higher flexibility in matching a query, we slightly generalized the tree query algorithm to enable also deletions of query nodes of degree 1 (leaves of the tree). We also included in QNet a heuristic that exploits the structure of the homology function to reduce the number of color coding iterations needed. In the following we describe this heuristic and the parameter setting employed in QNet.

*Restricted Color Coding.* We present a heuristic approach to color coding that tries to take advantage of queries whose protein members tend to have non-overlapping sets of homologs. First, we assign each query node a distinct *match color*, and choose  $N_{ins}$  additional *insertion colors*. Now, we color the network vertices using the following rule: For each network vertex  $v$ , if  $v$  is not homologous to any query protein, then assign it with a random insertion colors. Otherwise, toss a coin with probability  $p_t = \frac{N_{ins}}{k+N_{ins}}$ . If HEADS, choose a random insertion color for it, else if TAILS, assign it with a random color from the set of query nodes it is homologous to.

The probability  $P_s$  to obtain a colorful alignment subgraph is at least the probability that: (i) each aligned vertex is given a match color, and each inserted vertex is given an insertion color; and (ii) all colors are distinct. Let  $p_m$  be the probability that aligned vertices are colorful, and  $p_i$  be the probability that insertion vertices are colorful. Then

$$P_s = (1 - p_t)^k p_t^{n_i} p_i p_m = \left( \frac{k}{N_{ins} + k} \right)^k \left( \frac{N_{ins}}{k + N_{ins}} \right)^{N_{ins}} p_i p_m$$

where  $p_i \geq \frac{N_{ins}!}{N_{ins}^{N_{ins}}}$ . It remains for us to compute a lower bound for  $p_m$ . To this end, we form a graph on the set of query nodes, in which for every pair  $q, q'$  of query nodes, we add the edge  $(q, q')$  if there exists a network vertex  $v$  that is homologous to both. We then partition the query vertices into connected components  $Q_1, Q_2, \dots, Q_{k'}$ , and use the following bound:  $p_m \geq \prod_{u=1}^{k'} \frac{|Q_u|!}{|Q_u|^{|Q_u|}}$ . We expect  $p_m$  to be high since often query nodes are homologous to a single vertex. When the probability of success with restricted coloring is greater than the probability of success with the standard color coding (i.e.,  $\frac{(k+N_{ins})!}{(k+N_{ins})^{k+N_{ins}}}$ ), we use this procedure, and otherwise we use the standard color coding.

*Parameter Setting.* QNet involves several parameters controlling sequence similarity, insertion/deletion penalties, and the relative weights of edge- and node-terms. The current settings are as follows: we used blastp with an E-value threshold of  $10^{-7}$  to compute sequence similarity, and set  $h(q, v) = -\log(\text{E-value})$ . Interaction reliabilities  $p(u, v)$  are assigned using a logistic regression scheme based on the experimental evidences for the interactions, as described in [18]. We use  $w(u, v) = c \cdot r(u, v)$ , where  $c$  is chosen to ensure the same scale for the reliability and homology values. We allow at most two insertions and two deletions per query, i.e.,  $N_{ins} = N_{del} = 2$ . Indel penalties are set to  $\delta_d = \delta_i = -100$ . We empirically tested a range of penalties by querying perturbations of subtrees in the yeast network (see Section 5.1). A small set of queries were examined and the results did not change over the range as long as the net influence of a deletion or insertion were kept negative. In all runs reported below, the number of color coding iterations was set to ensure success probability  $\geq 0.99$ .

## 5 Experimental Results

To evaluate the performance of QNet we measure its running time and accuracy under various configurations. We start by applying QNet to query a set of synthetic trees in the PPI network of yeast, measuring its running time and accuracy. Next, we show examples of querying known yeast and human signal transduction pathways in the PPI network of fly. Finally, we apply QNet to query known yeast complexes in fly.

Protein-protein interaction data for yeast *S. cerevisiae* and fly *D. melanogaster* were obtained from the Database of Interacting Proteins (DIP) [20] (April 2005 download). The fly data was complemented by PPI interactions from [19] and by genetic interactions from FlyGRID (see also [18]). Altogether, the yeast network consists of 4,738 proteins and 15,147 interactions, and the fly network consists of 7,481 proteins and 26,201 interactions.

### 5.1 Synthetic Query Trees

To measure the running time and estimate the accuracy of QNet, we applied it to query the PPI network of yeast with a set of synthetic query trees. This set consists of 20 randomly chosen subtrees of sizes ranging from  $k = 5$  to  $k = 9$  from the yeast PPI network. Each query tree was perturbed with up to 2 node insertions and deletions, and by a pre-specified amount of point mutations in its proteins' sequences of average length  $\sim 500$ . QNet was applied to identify a match for each query tree.

The running time measurements were performed on a standard PC (2GHz, 1Gb). We find that the running time of QNet is a few seconds in all cases, reaching an average of 11 seconds for the largest tree queries with 9 nodes (Table 1). To measure the improvement in running time introduced by the restricted color coding heuristic, we applied QNet also without this heuristic. We find that restricted color coding significantly reduces the number of iterations required to

**Table 1.** Number of color coding iterations and timing statistics for QNet. The last two columns show the average time per query. The algorithm’s parameters are set as follows:  $N_{ins} = 2$ ,  $N_{del} = 2$ , and the probability of success is set to 0.99.

Query size ( $k$ )	#Iterations		Avg. time (sec.)	
	Standard color coding	Restricted color coding	Standard color coding	Restricted color coding
5	752	603	1.71	1.58
6	1916	917	6.36	4.73
7	4916	1282	20.46	6.24
8	12690	1669	61.17	9.08
9	32916	2061	173.88	11.03
10	85720	2509	1463	21.74
11	223990	2987	5501	41.39
12	1891868	4623	50455	97.93

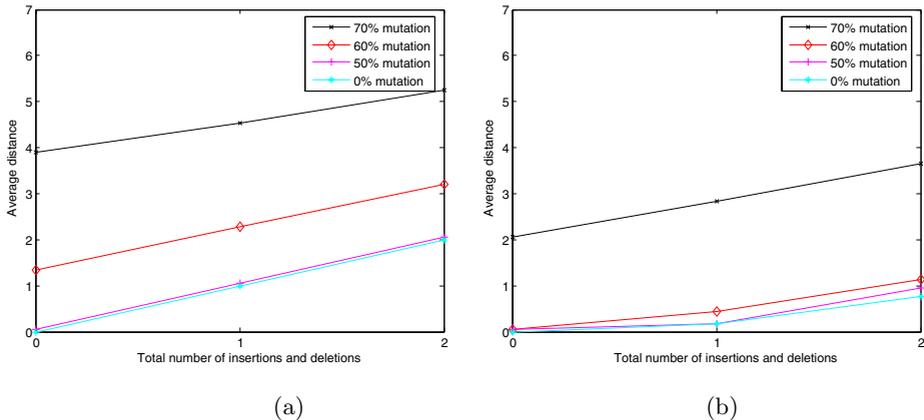
identify the optimal match, while the running time of each iteration remains similar. Overall, restricted color coding reduces the running time by an order of magnitude on average (Table 1). The running time of the algorithm is significantly affected by the number of insertions allowed. If no insertions are allowed, the average number of iterations required for queries of size 9 is less than 100. When increasing the number of allowed insertions to above 2, the restricted color coding heuristic becomes less effective (data not shown).

To evaluate the accuracy of the matched trees, we computed the symmetric difference between the protein set of a query and its match, termed their *distance* herein. The results show that when perturbing protein sequences in up to 60% of the residues, the average distance between the matched tree and the original tree is lower than 1 (Figure 3b). Moreover, we compared the accuracy of matches obtained by QNet to matches that are based only on best BLAST hits. We found that matches obtained by QNet are markedly more accurate than purely sequence-based matches, showing that the topology of the query tree carries important signal (Figure 3a). Evidently, the advantage of QNet over a sequence-based approach becomes more pronounced when the mutation rate increases.

## 5.2 Cross-Species Comparison of MAPK Pathways

The mitogen-activated protein kinase (MAPK) pathways are a collection of related signal transduction pathways, which play a critical role in mediating the cellular response to various toxic stresses [5]. The pathways are known to be conserved across species and, hence, serve as controlled tests to QNet.

We queried MAPK pathways from the KEGG database [12] in the PPI network of fly. The first pathway is a classical human MAPK pathway involved in cell proliferation and differentiation. Querying this pathway in fly resulted in detecting a known MAPK pathway involved in dorsal pattern formation (Figure 4a). Specifically, 6 out of the 8 matched proteins in the target are members of the known MAPK pathway in fly. Similar results were obtained by querying the yeast MAPK pathways from KEGG against the fly network. As an example, the top output for



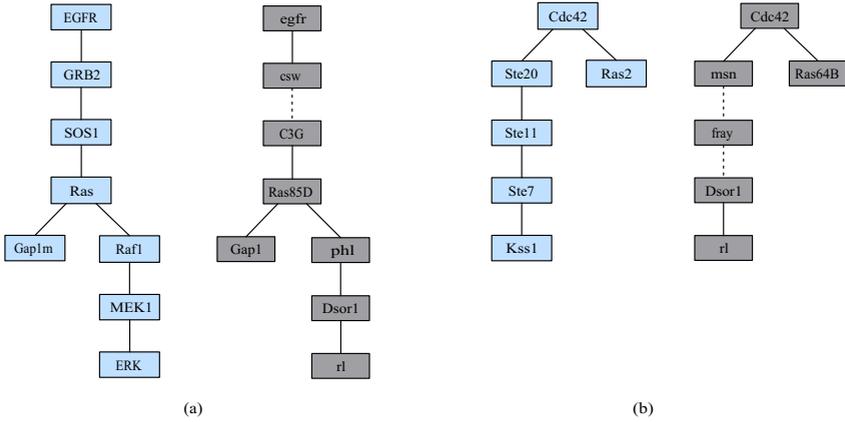
**Fig. 3.** The average distance of the matched tree from the original tree is plotted against the total number of insertions and deletions introduced to the query for 4 different mutation levels. (a) Performance of a sequence-based approach. (b) Performance of QNet.

the starvation response pathway query (Figure 4b) is a fly MAPK pathway with a putative MAPK cascade (fray,Dsor1,rl), which includes the GTPases Cdc42, Ras64b that are homologous to the two GTPases in the query. These results support the fidelity of QNet.

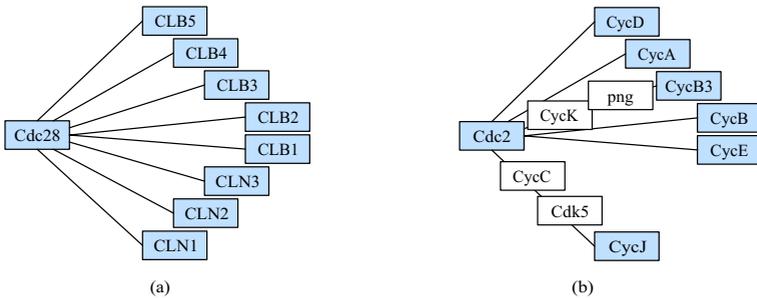
### 5.3 Cross-Species Comparison of Protein Complexes

As a large-scale validation of QNet we systematically queried known yeast protein complexes, obtained from the MIPS database [16,9], in the fly network, and tested the biological plausibility of the identified matches. We included all hand curated complexes in MIPS, which are considered a reliable data source, excluding complexes that were identified via high throughput measurements (category 550 in MIPS). Overall, we considered 94 complexes consisting of at least 4 proteins each. As MIPS does not contain information on the topology of the complexes, we mapped each complex to the yeast network and used the induced subnetworks as queries. More accurately, for each complex, we extracted an average of 40 random query trees of size in the range 3 – 8 from its induced subnetwork. We applied QNet to systematically query all of the induced query trees in fly. The resulting query matches were used to construct a *consensus match*, consisting of all proteins that appeared in at least half of the matches.

The biological plausibility of an obtained consensus matches was tested based on functional enrichment of their member proteins w.r.t. the fly gene ontology (GO) process annotation [2]. Specifically, let  $n(t)$  denote the number of genes in the consensus match that are annotated with term  $t$ . We compute the probability  $p(t)$  of obtaining a random set of genes, of the same size as the original pathway, with at least  $n(t)$  genes annotated with term  $t$ , assuming a hypergeometric



**Fig. 4.** Querying the fly network using (a) a human MAPK pathway, and (b) a yeast MAPK pathway induced by starvation, taken from the KEGG database [12]. Matched nodes appear on the same horizontal line. A dotted edge represents inserted proteins (not shown).



**Fig. 5.** (a) The MIPS Cdc28p complex. (b) The consensus match in fly. Matched nodes appear on the same horizontal line. Inserted proteins appear in white.

distribution. Having found a term  $t_0$  with minimal probability  $p(t_0)$ , we compute a  $p$ -value for the enrichment under term  $t_0$  by comparing  $p(t_0)$  with similar values computed for 10,000 random sets of genes. The latter  $p$ -values are further corrected for multiple match testing via the false discovery rate procedure [3].

36 of the yeast complexes resulted in a consensus match with more than one protein in fly. We find that 72% of these consensus matches are significantly functionally enriched ( $p < 0.05$ ). For comparison, we computed the functional enrichment of randomly chosen trees from the fly PPI network that have the same distribution of sizes and interactions scores as the consensus matches. We find that only 17% of the random trees are functionally enriched, and that the mean enrichment  $p$ -values is significantly lower for the true consensus matches (Wilcoxon rank test  $p$ -value  $< 6.5e - 9$ ).

Figure 5 illustrates the result of querying the Cdc28p complex. This complex is composed of cyclin-dependent kinases involved in regulating the cell cycle in yeast. The consensus match obtained in fly consists solely of cyclin-dependent kinases and significantly overlaps the cyclin-dependent protein kinase holoenzyme complex (GO:0000307).

## 6 Conclusion

Data sets of protein-protein interactions are increasingly common, and will continue to increase in number and complexity. In this paper, we address the problem of searching such data for specific pathways of interest. We provide efficient algorithms for querying trees and graphs of bounded treewidth within PPI networks. We implement the tree query algorithm, QNet, and demonstrate its efficiency and accuracy. QNet can handle queries of up to 9 proteins in seconds on current networks, and is shown to outperform sequence-based homology searches. More importantly, we use QNet to perform a large scale cross-species comparison of protein complexes, by querying known yeast complexes in the fly network. This comparison points to strong conservation between the two species.

While our work has helped in clarifying some algorithmic questions regarding efficient querying of biological networks, and has shown promising results in practice, it leaves many aspects open for future research. One important direction is the development of appropriate score functions to better identify conserved pathways. Research in this direction could gain from probabilistic models of network evolution [4,10]. A second important direction is the application of the methods developed here to queries of more general structure. This entails both the implementation and testing of a tool for querying bounded treewidth graphs, and the use of such a tool for querying arbitrary structures, perhaps in a way similar to that presented in Section 5.2.

## Acknowledgments

We thank Eyal Kaplan for critical reading of the manuscript. B.D. and V.B. were supported in part by the NSF grant CCF-0425926, T.S. was supported by the Tauber fund, and R.S. was supported by an Alon fellowship. This research was supported in part by a research grant from the Ministry of Science and Technology, Israel.

## References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.
2. M. Ashburner et al. The gene ontology consortium. gene ontology: Toll for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
3. Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Stat. Soc. B*, 57:289–300, 1995.

4. J. Berg, M. Lassig, and A. Wagner. Structure and evolution of protein interaction networks: A statistical model for link dynamics and gene duplications. *Bio. Med. Center Evolutionary Biology*, 4:51, 2001.
5. P. Dent, A. Yacoub, P. B. Fisher, M. P. Hagan, and S. Grant. Mapk pathways in radiation responses. *Oncogene*, 22(37):5885–5896, Sep 2003.
6. S. F and Z. R. Identifying active transcription factors and kinases from expression data using pathway queries. *Bioinformatics*, 21(Suppl 2):ii115–ii122, Sep 2005.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, 1979.
9. U. Guldener, M. Munsterkotter, M. Oesterheld, P. Pagel, A. Ruepp, H.-W. Mewes, and V. Stumpflen. MPact: the MIPS protein interaction resource on yeast. *Nucleic Acids Res*, 34(Database issue):436–441, Jan 2006.
10. E. Hirsh and R. Sharan. Identification of conserved protein complexes based on a model of protein network evolution. In *Fifth European Conference on Computational Biology (ECCB'06)*, 2006. To appear.
11. T. Ito, T. Chiba, and M. Yoshida. Exploring the yeast protein interactome using comprehensive two-hybrid projects. *Trends Biotechnology*, 19:23–27, 2001.
12. M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resource for deciphering the genome. *Nucleic Acids Res*, 32(Database issue):277–280, Jan 2004.
13. B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, B. R. Stockwell, and T. Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc Natl Acad Sci U S A*, 100(20):11394–9, 2003.
14. T. Kloks. *Treewidth: computations and approximations*. Springer-Verlag, 1994.
15. M. Mann, R. Hendrickson, and A. Pandey. Analysis ures of proteins and proteomes by mass spectrometry. *Annu. Rev. Biochem*, 70:437–473, 2001.
16. H. W. Mewes, D. Frishman, K. F. Mayer, M. Munsterkotter, O. Noubibou, P. Pagel, T. Rattei, M. Oesterheld, A. Ruepp, and V. Stumpflen. MIPS: analysis and annotation of proteins from whole genomes in 2005. *Nucleic Acids Res*, 34(Database issue):169–172, Jan 2006.
17. R. Y. Pinter, O. Rokhlenko, E. Yeger-Lotem, and M. Ziv-Ukelson. Alignment of metabolic pathways. *Bioinformatics*, 21(16):3401–8, 2005.
18. T. Shlomi, D. Segal, E. Ruppim, and R. Sharan. QPath: A Method for Querying Pathways in a Protein-Protein Interaction Network. *BMC Bioinformatics*, 7(199), 2006.
19. C. A. Stanyon, G. Liu, B. A. Mangiola, N. Patel, L. Giot, B. Kuang, H. Zhang, J. Zhong, and J. Finley, R. L. A Drosophila protein-interaction map centered on cell-cycle regulators. *Genome Biol*, 5(12):R96, 2004.
20. I. Xenarios, D. W. Rice, L. Salwinski, M. K. Baron, E. M. Marcotte, and D. Eisenberg. DIP: the database of interacting proteins. *Nucleic Acids Res*, 28(1):289–91, 2000.

## Appendix: A General Alignment Algorithm for Bounded Treewidth Queries

In Section 3.2 we described an algorithm for identifying optimal simple alignments of a bounded treewidth query graph. To generalize the algorithm to support deletions, we modify the mapping  $\sigma$  to allow mapping to ‘0’. To support insertions, we allow  $\sigma$  to map connected query nodes to non-connected graph vertices, and use additional  $N_{ins}$  color (as in Section 3.1).

Given the new definition of  $\sigma$ , the node term is modified as follows:

$$W^S(i, \sigma) = \delta_d |\{u \in X_i^N : \sigma(u) = 0\}| + \sum_{u \in X_i^N, \sigma(u) \neq 0} h(u, \sigma(u))$$

The edge term is more problematic as it depends on the subset of colors used for insertions, and requires some preprocessing. For a pair of vertices  $u, v \in V$  and a set of colors  $S \subseteq C - \{c(u), c(v)\}$ , we denote by  $W_P(u, v, S)$  the maximum weight of a path between  $u$  and  $v$  that visits the colors in  $S$ . Given a set of vertex pairs  $R = R(l) = \{(r_1^1, r_2^2), \dots, (r_l^1, r_l^2)\}$ , we define  $W_P(R, S)$  as the maximum weight of  $|R|$  simple paths between all vertex pairs that visit distinct colors from  $S$ :

$$W_P(R, S) = \max_{\substack{S^1, S^2, \dots, S^q \\ \uplus S^l = S}} \sum_{l=1}^q W_P(r_l^1, r_l^2, S^l)$$

In order to compute  $W_P(R, S)$  efficiently, we use the following recurrence:

$$W_P(R(l), S) = \max_{S' \subset S} [W_P((r_l^1, r_l^2), S') + W_P(R(l-1), S - S')]$$

Define  $E_i(\sigma)$  as the set of graph vertex pairs that are mapped from non-active edges in super-node  $i$ :

$$E_i(\sigma) = \{(u, v) \in E : (u', v') \in E_i^N, \sigma(u') = u, \sigma(v') = v\}$$

The edge term for super-node  $i$  under the mapping  $\sigma$  and colors  $S$ , is:

$$W^E(i, \sigma, S) = W_P(E_i(\sigma), S)$$

Finally, we modify the main recursion as follows:

$$W(i, \sigma, S) = \max_{\substack{S_1 \uplus S_2 = S, \\ S'_1 \subset S_1, S'_2 \subset S_2, \\ \sigma_1, \sigma_2}} \sum_{j=1}^2 [W(i_j, \sigma_j, S_j - S'_j) + W^S(i_j, \sigma_j) + W^E(i_j, \sigma_j, S'_j)]$$

To compute the running time of the preprocessing stage, note that  $W_P((u, v), S)$  can be pre-computed for all  $S$  in  $O(n^2 2^k)$  time. Therefore,  $W_P(E_i(\sigma), S)$  can be pre-computed in  $2^{O(k)} n^{t+1}$  time, and hence the total running time is  $2^{O(k)} n^{t+1}$ .