Topology-Free Querying of Protein Interaction Networks

Sharon Bruckner[1], Falk Hüffner[1], Richard M. Karp[2], Ron Shamir[1], and Roded Sharan[1]

[1] School of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel. {bruckner,hueffner,rshamir,roded}@tau.ac.il
[2] International Computer Science Institute, 1947 Center St., Berkeley, CA 94704, USA. karp@icsi.berkeley.edu

Abstract. In the network querying problem, one is given a protein complex or pathway of species A and a protein–protein interaction network of species B; the goal is to identify subnetworks of B that are similar to the query. Existing approaches mostly depend on knowledge of the interaction topology of the query in the network of species A; however, in practice, this topology is often not known. To combat this problem, we develop a topology-free querying algorithm, which we call Torque. Given a query, represented as a set of proteins, Torque seeks a matching set of proteins that are sequence-similar to the query proteins and span a connected region of the network, while allowing both insertions and deletions. The algorithm uses alternatively dynamic programming and integer linear programming for the search task. We test Torque with queries from yeast, fly, and human, where we compare it to the QNet topology-based approach, and with queries from less studied species, where only topology-free algorithms apply. Torque detects many more matches than QNet, while in both cases giving results that are highly functionally coherent.

1 Introduction

Sequence-based searches have revolutionized modern biology, serving to infer gene function, homology relations, protein structure, and more. In the last few years, there has been an effort to generalize these techniques to the network level. In a *network querying* problem, one is given a small subnetwork, corresponding to a pathway or a complex of interest. The goal is to identify similar instances in a large network, where similarity is measured in terms of sequence or interaction patterns.

The largest body of previous work on network querying concerns querying subnetworks across species. Kelley et al. [14] and later Shlomi et al. [29] devised fixed-parameter algorithms for querying linear paths within a protein–protein interaction (PPI) network. These algorithms were subsequently extended in the QNet software to allow searching for trees and bounded treewidth graphs [23]. A related work by Pinter et al. [21] presented a polynomial algorithm for detecting homeomorphic subtrees within a tree representing a collection of metabolic pathways. Another approach that relaxes the homomorphism but requires target and query nodes to agree in their neighborhood was given by Narayanan and Karp [18]. Sohler and Zimmer [30] 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. Yang and Sze [35] examine both query paths and the general case, but since their method is based on exhaustive enumeration, it can also handle only small queries.

Another line of work on network querying has been the search for small motifs that are defined in terms of the functional attributes of their member proteins and the interactions among them. Lacroix et al. [16] suggested a branch-and-bound approach for finding connected subgraphs whose vertex set matches a query, which they applied to small queries (of size 2–4) only. Betzler et al. [4] gave a fixed parameter algorithm for the latter problem and some extensions of it. An additional heuristic solution was offered by Zheng et al. [37] to a similar problem, in the context of querying metabolic networks. Finally, Ferro et al. [9] presented the GraphFind algorithm, which utilizes fast heuristics for subgraph isomorphism to identify approximate matches of queries within a collection of networks.

A limitation of the approaches above (except for [16],[4]) is that they rely on precise information on the interaction pattern of the query pathway. However, often this information is missing. For example, hundreds of protein complexes have been reported in the literature for yeast [27], human [25], and other species. However, for most of these complexes no information exists on their interaction patterns [36], motivating a topology-free approach for the querying problem.

Here we devise Torque (TOpology-free netwoRk QUErying), a novel approach for network querying that does not rely on knowledge of the query topology. The input to our method is a set of proteins, representing a protein complex or pathway of interest and a network in which the search is to be conducted. The goal is to find matching sets of proteins that span connected regions in the network. The corresponding theoretical problem that we study is searching a colored graph for connected subgraphs whose vertices have distinct given colors. We provide fixed-parameter algorithms that are based on the color-coding paradigm [1] and dynamic programming (DP) for several variants of this problem. In addition, we provide an integer programming (ILP) formulation of it. That formulation includes a novel way to describe subgraph connectivity constraints, which can be useful in other problems as well. The methods can handle edge weights, insertions of network vertices (that do not match any query protein), and deletions of query nodes. By using DP and ILP approaches, we can query complexes of all sizes within current networks in reasonable time.

We applied Torque to query about 600 known complexes of size 4–25 from a variety of species in the PPI networks of yeast, fly and human. We tested our algorithm both on queries from species for which a PPI network is available, where we compared it to the QNet [23] topology-based approach, and on queries from less studied species, where only topology-free algorithms apply. Torque detected many more matches than QNet, while in both cases giving results that are highly functionally coherent.

Due to space limitations, some proofs are omitted. They will be included in a full version of this manuscript.

2 Algorithms

Let $G = (V, E)$ be a PPI network where vertices represent proteins and edges correspond to PPIs. Denote $|V| = n$ and $|E| = m$. For a vertex v, let $N(v)$ denote the set of its neighbors, i. e., $N(v) = \{u : (u, v) \in E\}$. For two disjoint sets S_1 and S_2, we write $S_1 \uplus S_2$ for their union $S_1 \cup S_2$. We denote by $G[K]$ the subgraph of G induced by the vertex set K.

Given a set of colors $C = \{1, 2, \ldots, k\}$, a *coloring constraint* function $\mathit{\Gamma} : V \rightarrow 2^C$ associates with each $v \in V$ a subset of colors $\mathit{\Gamma}(v) \subseteq C$. For $S \subseteq C$, we define a subset $H \subseteq V$ as *S-colorful* if $|H| = |S|$ and there is a function c that assigns each $v \in H$ a color from $\mathit{\Gamma}(v)$, such that there is exactly one vertex in H of each color in S. The basic problem that we study is the following:

*Problem 1 (C-*COLORFUL CONNECTED SUBGRAPH*).* Given a graph $G = (V, E)$, a color set C, and a coloring constraint function $\mathit{\Gamma} : V \rightarrow 2^C$, is there a connected subgraph of G that is C-colorful?

This problem was shown to be NP-complete by Fellows et al. [8], even for the case of trees of maximum degree 3. Here we provide fixed-parameter tractable algorithms for several variants of this problem, where the parameter is the size of the query complex. A problem is fixed-parameter tractable with respect to a parameter k if an instance of size n can be solved in $O(f(k) \cdot n^{O(1)}$ time, where f is an arbitrary function. Thus, fixed-parameter algorithms allow solving relatively large instances of NP-hard problems exactly [19], as long as the parameter value is modest.

2.1 Single color constraints

In the first variant of the problem, we consider only coloring constraint functions that associate each $v \in V$ with a single color. In this case, the input is a graph where each vertex is assigned a color from C, and we aim to find a connected subgraph having exactly one vertex of each color.

Since every connected subgraph has a spanning tree, it suffices to look for colorful trees. This problem has been studied by Scott et al. [26] in another context, as well as by Kalaev et al. [13], Betzler et al. [4]. For completeness, we provide a dynamic programming (DP) algorithm, which is the unweighted version of the algorithm given by Scott et al. [26]. We construct a table B with rows corresponding to vertices and columns corresponding to subsets $C' \subseteq C$. We define $B(v, S)=$ *True* if there exists in G a subtree rooted at v that is S-colorful,

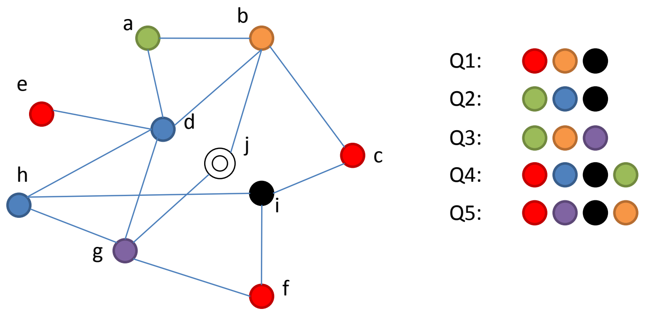

Fig. 1. Network query problems. Left: the network, where vertex j is non-colored. Right: queries. For the basic problem disallowing indels, $Q1$ is solved by $\{c, b, i\}$, while $Q2$ and $Q4$ have no solution. When allowing a single arbitrary insertion, $Q2$ has solution $\{a, d, h, i\}$ and $Q4$ has the solution $\{a, b, c, d, i\}$. When allowing a single special insertion, $Q3$ has the solution $\{a, b, g, j\}$. When allowing one deletion, $Q2$ has the solutions $\{a, d\}$, $\{i, f\}$. When allowing repeated nodes and no indels, $Q5$ has the solution $\{b, c, i, f, g\}$.

and *False* otherwise. For $S = \{\gamma\}$ and $v \in V$ we initialize $B(v, \gamma) =$ *True* iff $\Gamma(v) = \{\gamma\}$. Other entries of B can be computed using the following recurrence:

$$B(v, S) = \bigvee_{\substack{u \in N(v) \\ S_1 \uplus S_2 = S \\ \Gamma(v) \in S_1, \Gamma(u) \in S_2}} B(v, S_1) \wedge B(u, S_2), \tag{1}$$

The algorithm runs in $O(3^k m)$ time[3]. One can easily generalize (1) to the weighted case, where each edge is assigned a weight, and the heaviest tree is sought.

Insertions and Deletions. Exact matches are often impossible due to evolutionary variation and noise in the data. Hence, we would like to allow deletions of query proteins that cannot be matched and insertions of network proteins that assist in connecting matched vertices. Deletions can be directly handled by the DP algorithm: If no C-colorful solution was found, then $B(v, C) =$ *False* for all v. Allowing up to N_{dels} deletions can be done by scanning the entries of B. If there exists $\hat{C} \subseteq C$ such that $|\hat{C}| \geq |C| - N_{dels}$, and $B(v, \hat{C}) =$ *True* then a valid solution exists.

When allowing insertions, there are several problem variants to consider (see Figure 1). In the first variant, some network vertices are not assigned a color, and only non-colored vertices can be inserted. For convenience, assign non-colored vertices the color 0. Let us call such insertions *special*.

[3] It can be further reduced to $O(2^k m)$ using the techniques of Björklund et al. [5]; however, this version cannot be generalized to the weighted case, so we do not use it in the following.

Definition 1. *An* S-colorful solution allowing j special insertions *is a connected subgraph $H \subseteq G$, where $\exists H' \subseteq H$ such that $V(H')$ is S-colorful and all other vertices of H are non-colored.*

An obvious extension of the DP algorithm to handle up to N_{ins} special insertions is based on the color-coding paradigm of Alon et al. [1]: Randomly color the non-colored vertices with N_{ins} new colors and use DP to look for colorful trees. This procedure is repeated a sufficient number of times to ensure that every tree is colorful with high probability. However, the running time increases by a factor of $(3e)^{N_{\text{ins}}}$. We provide a more efficient solution below.

Theorem 1. *Finding a C-colorful connected subgraph with up to N_{ins} special insertions can be solved in $O(3^k m N_{ins})$ time.*

Proof. We extend the DP table to represent also the number of special insertions used in an intermediate solution. Formally, $B(v, S, j)$ iff there is an S-colorful subtree rooted at v that allows j special insertions, and j is the minimal number of insertions possible. Here j ranges between 0 and N_{ins}. We initialize the table by setting all entries to *False*, except: (i) For $\gamma \neq 0$, $B(v, \{\gamma\}, 0)$ iff $\varGamma(v) = \gamma$; and (ii) if $\varGamma(v) = 0$, $B(v, \emptyset, 1)$. Entries for which $|S| \geq 1$ and $j > 0$ are then computed using the following recurrence:

$$B(v,S,j) = \Big[\bigvee_{\substack{u \in N(v) \\ S_1 \uplus S_2 = S \\ j_1 + j_2 = j}} B(v, S_1, j_1) \wedge B(u, S_2, j_2) \Big] \wedge \forall j' < j : \neg B(v, S, j') \tag{2}$$

We prove correctness by induction on $|S|$ and j. The cases $j = 0$ and $S = \emptyset$ are immediate. Therefore, consider $j > 0$ and as the first case $S = \{\gamma\}$ (i. e., $|S| = 1$). By definition:

$$\begin{gathered} B(v, \{\gamma\}, j) \iff \exists u \in N(v), S_1, S_2, j_1, j_2 : \\ B(v, S_1, j_1),\ B(u, S_2, j_2),\ S_1 \uplus S_2 = \{\gamma\},\ j_1 + j_2 = j,\ \forall j' < j : \neg B(v, \{\gamma\}, j') \end{gathered}$$

Assuming there are u, S_1, S_2, j_1, j_2 as above, then S_1 cannot be $\{\gamma\}$ since $\forall j' < j : \neg B(v, \{\gamma\}, j')$. It follows that $S_1 = \emptyset$ and $S_2 = \{\gamma\}$, implying that $\varGamma(v) = 0$, $j_1 = 1$, and $j_2 = j - 1$ (see initialization of B). By the induction hypothesis on j, $B(u, \{\gamma\}, j - 1)$ implies that there exists a tree T rooted at u having one vertex colored γ and a minimal number of $j - 1$ non-colored vertices. Clearly, $v \notin T$. Otherwise, there will be a tree T' rooted in v having one vertex colored γ and $j' < j$ special vertices, in contradiction to the minimality of j. Since $u \in N(v)$, then $T \uplus \{v\}$ is a tree having one vertex colored γ and j non-colored vertices, as desired.

It remains to handle the case where $|S| > 1$. By definition:

$$\begin{gathered} B(v, S, j) \iff \exists u \in N(v), S_1, S_2, j_1, j_2 : \\ B(v, S_1, j_1),\ B(u, S_2, j_2),\ S_1 \uplus S_2 = S,\ j_1 + j_2 = j,\ \forall j' < j : \neg B(v, S, j') \end{gathered}$$

Suppose such u, S_1, S_2, j_1, j_2 exist. Then by the induction hypothesis, there is a tree T_v rooted at v that is S_1-colorful and contains a minimal number j_1 of special vertices. Similarly, there is a tree T_u rooted at u that is S_2-colorful and contains a minimal number j_2 of special vertices. T_u and T_v are clearly disjoint: Otherwise, there would be another tree T' rooted at v which is S-colorful and contains $j' < j_1 + j_2$ special vertices, in contradiction to $\neg B(v, S, j')$. Since $u \in N(v)$, the union of these trees is $(S_1 \uplus S_2)$-colorful and has $j_1 + j_2$ special vertices, as desired.

To achieve the stated running time, we maintain an auxiliary function $t(v, S)$ which is set to j when for the first time $B(v, S, j)$ is true for some j. In the recursion (2), we replace the condition $j_1 + j_2 = j$ by $t(v, S_1) + t(u, S_2) = j$. Since the table is $(N_{\text{ins}} + 1)$ times the size of the table in the basic case, the running time increases by a factor of N_{ins} compared to the basic case. $\square$

In a second variant of insertion handling, any vertex can be inserted (rather than only non-colored ones). We solve this variant by using the algorithm for the problem with special insertions as a black box. Instead of running the algorithm on the input graph G, we run it on an auxiliary graph $G' = (V', E')$, which is constructed as follows: Add a non-colored copy v^0 for each $v \in V$, and set $E' = E \cup \{(v^0, u) \mid (v, u) \in E\} \cup \{(v^0, u^0) \mid (v, u) \in E\}$. This variant has a running time of $O(N_{\text{ins}} 3^k m)$.

2.2 Multiple color constraints

We now turn to the more general case, where a color constraint function can associate each vertex with a set of colors and not just a single color. This problem arises when a network vertex protein is homologous to several query proteins. Betzler et al. [4] gave a fixed-parameter algorithm for the problem, where the running time is increased by a factor of $(2e)^k$ compared to the case of single color constraints. Here we give an alternative fixed-parameter algorithm (coupled with some speedup heuristics). The basic idea is to reduce the problem to the single color case by randomly choosing a single valid color for every vertex. Our main effort is in computing an upper bound on the number of coloring iterations needed.

Define a *color graph* to be a bipartite graph $B = (V, C, E)$ where V is the set of network vertices, C is the set of colors and $(v, c) \in E \iff c \in \mathit{\Gamma}(v)$. Consider a possible match to the query; for clarity, we assume that this match does not contain insertions or deletions. Then we can prove the following bound:

Theorem 2. *The probability for a subset of vertices of size k to become colorful in a random coloring is at least $\frac{1}{k!}$.*

The above theorem implies an overall running time of $O(k! 3^k m N_{\text{ins}}^2)$ in the case of multiple color constraints. However, this bound is excessive in many instances, for the following reason. Let V' be a colorful set of vertices. Following [23], define the *constraint graph* $G(V')$ as follows: the vertices are the colors, and an edge exists between two colors γ_1, γ_2 if there is a vertex v in V' such that $\gamma_1, \gamma_2 \in \Gamma(v)$. The resulting graph is then partitioned into connected components $P_1, P_2, \ldots, P_s$. This partition induces a partition of the colored network vertices into sets $Q_1, Q_2, \ldots, Q_s$, where all the vertices of Q_i can be colored only by colors from P_i. The expected number of iterations required for a P_i-sized subset of Q_i to become colorful is bounded by $|P_i|!$, and thus the number of iterations required for a solution of size k to become colorful is bounded by $\prod_{i=1}^{s} |P_i|!$. Therefore, since the number of iterations required by the algorithm is bounded by the number of iterations required before V' becomes colorful, the expected number of iterations of the algorithm is also bounded by the same product. Note, however, that the bound cannot be precomputed, although an upper bound can be obtained by taking $V' = V$.

We can reduce this upper bound using the following two rules: (i) If for some i, the product of all color degrees in Q_i is smaller than $|P_i|!$, then it is beneficial to exhaustively enumerate all possible colorings of Q_i. (ii) By Hall's Theorem [17], if a graph has a perfect matching and its minimum degree is d, then it has at least $d!$ perfect matchings. Therefore, if the minimal color degree in Q_i is d, $\frac{|P_i|!}{d!}$ random iterations suffice.

2.3 An integer programming formulation

In this section we provide an ILP formulation of the network querying problem, allowing us to employ industrial solvers that on certain instances are faster than DP. Formally, the problem that we aim to solve using the ILP is Problem 1 (C-COLORFUL CONNECTED SUBGRAPH) with exactly N_{ins} arbitrary insertions. Further, we are given edge weights $\omega : E \to \mathbb{Q}$ and wish to find a vertex subset $K \subseteq V$ of size $t := k + N_{\text{ins}}$ that maximizes the total edge weight $\sum_{(v,w)\in E, v,w \in K} \omega_{vw}$.

We declare binary variables $\{c_v : v \in V\}$ that express whether a vertex v is selected into the complex K. It is easy to give constraints that ensure correct coloring; the difficulty is in expressing the connectivity. The idea is to find a flow[4] with $t-1$ selected vertices as sources of flow 1, and a selected sink r that drains a flow of $t-1$, while disallowing flow between non-selected vertices. We use the following variables:

$\{c_v : v \in V\}, c_v \in \{0,1\}$	vertex v is selected ($v \in K$)	(3)
$\{e_{vw} : (v,w) \in E, v < w\}, e_{vw} \in \{0,1\}$	edge (v,w) is in $G[K]$	(4)
$\{r_v : v \in V\}, r_v \in \{0,1\}$	vertex v is the sink	(5)
$\{f_{vw}, f_{wv} : (v,w) \in E\}, f_{vw}, f_{wv} \in \mathbb{Q}$	flow from v to w/w to v	(6)
$\{g_{v\gamma} : v \in V, \gamma \in \Gamma(v)\}, g_{v\gamma} \in \{0,1\}$	vertex v has color γ	(7)

[4] That is, a function $f : V \times V \to \mathbb{Q}$ that satisfies skew symmetry ($\forall v, w \in V : f(v,w) = -f(w,v)$) and flow conservation ($\sum_{w \in V} f(v,w) = 0$) for all vertices v except sources and sinks; see e. g. Cormen et al. [7] for an introduction on flows.

and the following constraints

$$\sum_{v \in V} c_v = t \tag{8}$$

$$\sum_{v \in V} r_v = 1 \tag{9}$$

$$e_{vw} \leq \frac{1}{2} c_v + \frac{1}{2} c_w \quad \forall (v,w) \in E \tag{10}$$

$$f_{vw} = -f_{wv} \quad \forall (v,w) \in E \tag{11}$$

$$\sum_{w \in N(v)} f_{vw} = c_v - t r_v \quad \forall v \in V \tag{12}$$

$$f_{vw}, f_{wv} \leq (t-1) e_{vw} \quad \forall (v,w) \in E \tag{13}$$

$$\sum_{\gamma \in \Gamma(v)} g_{v\gamma} \leq 1 \quad \forall v \in V \tag{14}$$

$$\sum_{v \in V} g_{v\gamma} = 1 \quad \forall \gamma \in C \tag{15}$$

$$g_{v\gamma} \leq c_v \quad \forall v \in V, \gamma \in \Gamma(v) \tag{16}$$

with the objective

$$\text{maximize} \sum_{(v,w) \in E} \omega_{vw} e_{vw}. \tag{17}$$

The ILP can be easily adapted to allow N_{del} deletions by changing the constraint (15).

3 The implemented algorithm

We implemented a pipeline for querying a complex given as a set of proteins from a source species, in the PPI network of a target species. For each complex, Torque is applied with increasing number of allowed indels until a match is found or a pre-specified bound on the number of indels is reached. We use the multiple colors per vertex model and arbitrary insertions. The possible matches for the query in each network sub-component (see below) are assigned a score based on edge weights, and the highest scoring match is finally output. We will now describe the stages of the algorithm, the scoring scheme, and the parameters we used for our testing.

Preprocessing. A protein complex is specified as a set of proteins. We associate a distinct color with each query protein and define a corresponding coloring constraint function. Each vertex in the target network is associated with a subset of colors corresponding to the query proteins it is sequence-similar to. In practice, only 5% of the vertices on average are associated with one or more colors. The rest are treated as non-colored.

While some of the non-colored vertices can be used as insertion vertices, many are too far from any colored vertex to be feasible insertions under the given upper bound N_{ins}. Let v be a non-colored vertex, and let $d_0(u, v)$ be the the length (number of edges) of the shortest path between u and v where every vertex on the path is required to be non-colored. We keep v if there are colored vertices u_1, u_2 such that $d_0(u_1, v) + d_0(u_2, v) \leq N_{\text{ins}} + 1$, and the corresponding paths are vertex-disjoint. Otherwise, we remove v from the network. On the networks and complexes that we tested (see below), subnetworks containing only colored vertices are usually of size less than 50, those allowing 1–2 insertions have 200–1000 vertices, and those allowing more insertions typically cover up to 99% of the network.

After computing the current subnetwork to search in, we partition it into its connected components and search in each one independently. We call a component *feasible* if the color constraints of its vertices contain at least $k - N_{dels}$ colors of the query. Next, we process feasible connected components of increasing size, searching for the highest scoring matched complex using the DP or the ILP methods. We increase the number of indels, generating larger connected components, until a solution is found that contains the minimal number of insertions and deletions, where insertions are preferred over deletions, as they can be better attributed to incomplete data.

Running the query algorithms. When querying a connected component, its unique properties dictate which of our two methods will find a match more efficiently. As a rule of thumb, when the number of vertices is very close to the number of colors k, and k is large, the ILP algorithm is preferable, since we have observed that its running time is not as sensitive to k, while the color-coding algorithm has an exponential dependency. Empirically, we apply the ILP algorithm whenever $2^{n-k} < 3^k$, where n is the size of the connected component. This condition was satisfied in about 2/3 of the queries we used. For the DP algorithm, we used the multiple colors per vertex model, generating coloring assignments for the vertices using the bounds described in Section 2.2, thus reducing the number of iterations required.

Scoring. We score a set of proteins matching a query using the approach of Sharan et al. [28]. Briefly, a match is assigned a likelihood ratio score, which measures its fit to a protein complex model (assuming that every two proteins in a complex should interact with high probability, independently of all other pairs) vs. the chance that its connections in the target network arise at random. To accommodate for information on the reliability of interactions, the interaction status of every vertex pair is treated as a noisy observation, and its reliability is combined into the likelihood score.

When applying the DP algorithm, we output the highest scoring tree rooted at each vertex. Then, for each such solution tree, we compute the score of the subgraph that is induced by its vertices, taking into account edges and non-edges, to produce a final score for this vertex set. For the ILP, we improve the running time by assuming that all negatively weighted vertex pairs (v, w) (non-edges in the network) have the same weight penalty, thus avoiding to introduce integer variables e_{vw}.

Parameter setting. Our tests were performed using the following set of parameters. We queried complexes of size 4–25. Query and network protein sequence similarities were evaluated using BLAST. For a vertex v and a color γ, we let $\gamma \in \Gamma(v)$ if the BLAST E-value obtained by comparing the sequences of v and the query protein corresponding to γ was less than 10^{-7}. For each complex, we allowed TORQUE to run at most 4000 seconds, and took the best solution up to that point. We set the number of allowed insertions and deletions to 2 of each for small complexes (size < 7), 3 of each for medium sized complexes (size 8–14), and 4 of each for larger complexes. We intend to explore the robustness of these parameters in the journal version.

The algorithms were implemented with Python 2.5.2; for the ILP we used CPLEX 11.0.1. The test machine was a 3 GHz Intel Xeon (only one CPU was used) with 8 G of memory, running Debian GNU/Linux 4.0.

4 Experimental results

We applied TORQUE to query protein complexes within the three largest eukaryotic PPI networks available to date: yeast (5430 proteins, 39936 interactions), fly (6650 proteins, 21275 interactions) and human (7915 proteins, 28972 interactions). As queries, we used six collections of protein complexes from different species: yeast, fly, human, bovine, mouse, and rat. The first three served us to validate our algorithm and compare it to the state-of-the-art QNet algorithm [23][5]. The last three, for which no large-scale PPI information exists, allowed us to explore the power of the algorithm in querying protein complexes for which no topology information is available. In the following we describe the data, evaluation measures and the results obtained.

Data acquisition. For yeast, fly and human we obtained up-to-date PPI data, gathered from recently published papers [31, 24, 32, 11, 15, 22] and from public databases [34, 10, 20]. High-throughput mass spectrometry data [11, 15] was translated into binary PPIs using the spoke model [2]. Yeast complexes were downloaded from SGD [27] (Macromolecular Complex GO-Slim category). Fly complexes were obtained using the AmiGo [12] browser to collect all proteins annotated with GO:0043234 (protein complex). The complexes for all mammals (human, mouse, rat, bovine) were downloaded from the CORUM website [25].

Quality evaluation. To evaluate the quality of the matches, we used two measures: *functional coherence* and *specificity*. The first measure reports the percent of matches that are significantly functionally coherent with respect to the Gene

[5] A comparison to GraphFind [9] was not feasible, since its interface does not allow automated execution of the more than 600 queries.

Ontology (GO) [33] annotation. Note that while the query is functionally coherent, the reported matches may not be so due to permissive homology matching and the noise in the PPI data. To compute the functional coherence of a match, represented as a set of proteins, we used the GO TermFinder [6] tool. The p-values returned by the tool were further corrected for multiple match testing using the false discovery rate (FDR) procedure [3].

The second measure reports the specificity of the suggested solution, i. e., the fraction of matches that significantly overlap with a known protein complex. The significance of the overlap was evaluated using the hypergeometric distribution. The resulting p-value was compared to those obtained on 100 random sets of proteins of the same size to produce an empirical p-value. Those p-values were FDR-corrected for multiple testing. In the specificity computation we focused on matches that had a non-zero overlap with the collection of complexes to which they were compared. We also report separately those *novel matches* that had no overlap with known complexes. Although it is possible that some of these non-overlapping matches are false positives, we believe that the high percentage of specific matches indicate that some - or most - of these are indeed novel complexes.

Comparison to QNet. Our first set of experiments focused on the yeast, fly and human networks and protein complex collections. For each of the three species, we queried its complexes in the networks of the other two species. As large-scale networks are available in this setting, we could compare ourselves to the QNet algorithm [23], which was designed to tackle topology-based queries. While exact topology for the query complexes is mostly unknown, QNet infers it by projecting the complexes onto the corresponding network. This results in a set of possible spanning trees for the complex that are hence provided to QNet as inputs. This makes QNet very dependent on the quality of the source network, in addition to the usual dependence on the quality and completeness of the target network. We used the original QNet code with the same machine setup and parameters as our algorithm: sequence similarity, insertions and deletions, and time limits.

A striking difference between TORQUE and QNet can be seen from the results in Figure 2—out of 433 feasible queries overall, TORQUE detected matches for 311, while QNet found matches for 114 only. As we show below, this 45% gain in sensitivity did not harm the specificity of the results.

Next, we turned to evaluate the results using the functional coherence and specificity measures described above. The results for the three data sets are summarized in Table 1. As evident from the table, even though TORQUE matched many more queries, its results exhibit higher functional coherence and similar specificity levels.

Topology-free queries. A unique characteristic of TORQUE is its ability to query protein complexes for which a topology is not known. Here we apply our algorithm to query, for the first time, sets of protein complexes of mouse (59 complexes), rat (55) and bovine (10) – species for which no large scale PPI data

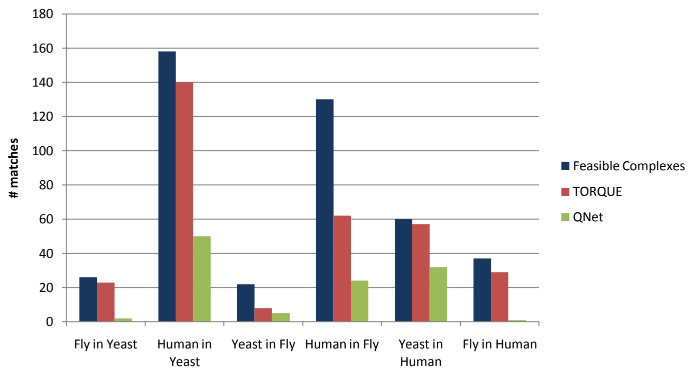

Fig. 2. Comparison of number of matches for Torque and QNet.

Table 1. Number and percentage of matches found that pass a quality significance threshold of 0.05.

		Functional coherence		Specificity		Novel matches	
Network	Complex	Torque	QNet	Torque	QNet	Torque	QNet
Yeast	Fly	23 (100%)	2 (100%)	19 (82%)	2 (100%)	7	0
	Human	134 (95%)	49 (98%)	119 (85%)	47 (94%)	8	2
Fly	Yeast	8 (100%)	3 (60%)	8 (100%)	4 (80%)	1	0
	Human	56 (90%)	21 (87%)	62 (100%)	23 (95%)	22	5
Human	Yeast	48 (84%)	25 (78%)	43 (75%)	23 (71%)	8	6
	Fly	21 (72%)	0 (—)	21 (72%)	0 (—)	7	0
Total		290	100	272	99	46	13

Table 2. Statistics of querying protein complexes for which no topology information is available.

Network	Complex	#Feasible	#Matches	Functional coherence	Specificity	Novel matches
Yeast	Bovine	4	4	4	4	0
	Mouse	17	17	16	13	1
	Rat	23	20	19	9	6
Fly	Bovine	3	0	-	-	-
	Mouse	14	7	0	1	6
	Rat	34	21	17	7	14
Human	Bovine	4	4	2	1	0
	Mouse	48	46	32	24	6
	Rat	44	43	32	24	4
Total		168	162	122	83	37

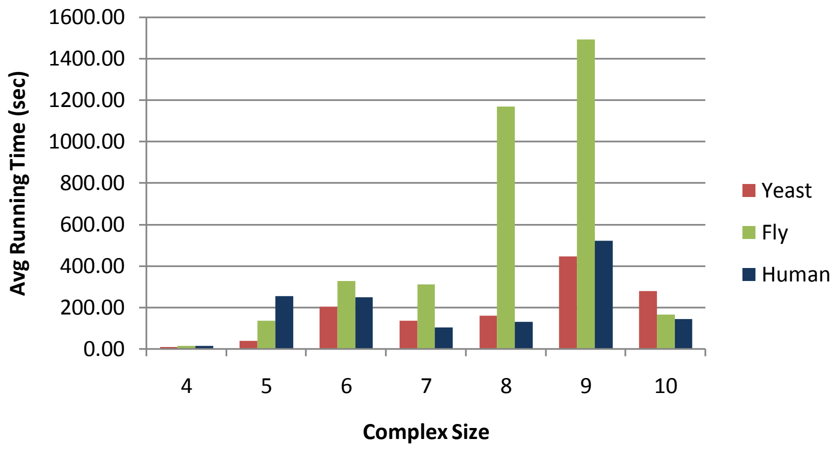

Fig. 3. Running time as a function of complex size.

are currently available. In Table 2, we present the results of querying these complexes within the networks of yeast, fly, and human. As evident from the table, more than 95% of the feasible queries had a match, and the majority of the matches were functionally enriched or matched a known complex.

Running time. The running time of TORQUE depends on many factors: complex size, number of homologs for each query protein, and the size of the connected component being tested. Figure 3 shows the running time of TORQUE for complexes of size up to 10, for those instances that the algorithm finished its processing on within the 1 hour time limit. Queries of size > 10 were handled by the ILP algorithm, whose average running time was 0.29 seconds for the queries whose processing was completed. Out of the total 624 feasible queries of all sizes, the processing of 122 was not completed in time.

5 Conclusions

We presented a tool for querying protein complexes for which no topology information is available within a PPI network. Compared to a topology-based approach, our program produces many more matches that are highly functionally enriched. Thus, our tool seems practical for a wide range of network query tasks, in particular involving species with sparse data. It would be interesting to examine matches for biological significance, in particular those that do not overlap any known complex and may suggest unknown complexes of the target species. This may also allow us to design a fine-tuned cost model, that takes costs of insertions, deletions, and particular protein mappings into account.

Acknowledgments We thank Noga Alon for his help in analyzing the case of multiple color constraints. We thank Banu Dost for providing us with the QNet code, and Nir Yosef for providing the PPI networks. R. Shamir and R. Sharan were supported in part by the Israel Science Foundation (grant no. 385/06). F. Hüffner was supported by a postdoctoral fellowship from the Edmond J. Safra Bioinformatics Program at Tel Aviv University.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color coding. *Journal of the ACM*, 42: 844–856, 1995.

[2] G. D. Bader and C. W. Hogue. Analyzing yeast protein-protein interaction data obtained from different sources. *Nature Biotechnology*, 20(10):991–997, 2002.

[3] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)*, 57(1):289–300, 1995.

[4] N. Betzler, M. R. Fellows, C. Komusiewicz, and R. Niedermeier. Parameterized algorithms and hardness results for some graph motif problems. In *Proc. 19th CPM*, volume 5029 of *LNCS*, pages 31–43. Springer, 2008.

[5] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. 39th STOC*, pages 67–74, New York, 2007.

[6] E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J. M. Cherry, and G. Sherlock. GO::TermFinder—open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, 20(18):3710–3715, 2004.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[8] M. R. Fellows, G. Fertin, D. Hermelin, and S. Vialette. Borderlines for finding connected motifs in vertex-colored graphs. In *Proc. 34th ICALP*, volume 4596 of *LNCS*, pages 340–351. Springer, 2007.

[9] A. Ferro, R. Giugno, M. Mongiovì, A. Pulvirenti, D. Skripin, and D. Shasha. Graphfind: enhancing graph searching by low support data mining techniques. *BMC Bioinformatics*, 9 Suppl 4:1471–2105, 2008.

[10] FlyBase-Consortium. The FlyBase database of the drosophila genome projects and community literature. *Nucleic Acids Research*, 31(1):172–175, 2003.

[11] A. C. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. J. Jensen, S. Bastuck, B. Dumpelfeld, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, 2006.

[12] GO Consortium. Amigo. http://amigo.geneontology.org/, Sept. 2008.

[13] M. Kalaev, V. Bafna, and R. Sharan. Fast and accurate alignment of multiple protein networks. In *Proc. 12th RECOMB*, volume 4955 of *LNCS*, pages 246–256. Springer, 2008.

[14] B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 32(Web Server issue), July 2004.

[15] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

[16] V. Lacroix, C. Fernandes, and M. Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):360–368, 2006.

[17] L. Lovász and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland, Amsterdam, 1986.

[18] M. Narayanan and R. M. Karp. Comparing protein interaction networks via a graph match-and-split algorithm. *Journal of Computational Biology*, 14(7):892–907, 2007.

[19] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[20] S. Peri, J. D. Navarro, R. Amanchy, T. Z. Kristiansen, C. K. Jonnalagadda, V. Surendranath, V. Niranjan, B. Muthusamy, T. K. Gandhi, M. Gronborg, et al. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Research*, 13(10): 2363–2371, 2003.

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

[22] T. Reguly, A. Breitkreutz, L. Boucher, B. J. Breitkreutz, G. C. Hon, C. L. Myers, A. Parsons, H. Friesen, R. Oughtred, A. Tong, et al. Comprehensive curation and analysis of global interaction networks in Saccharomyces cerevisiae. *Journal of Biology*, 5(4):11, 2006.

[23] R.Sharan, B. Dost, T. Shlomi, N. Gupta, E. Ruppin, and V. Bafna. Qnet: A tool for querying protein interaction networks. *Journal of Computational Biology*, 15(7):913–925, 2008.

[24] J. F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein-protein interaction network. *Nature*, 437(7062):1173–1178, 2005.

[25] A. Ruepp, B. Brauner, I. Dunger-Kaltenbach, G. Frishman, C. Montrone, M. Stransky, B. Waegele, T. Schmidt, O. N. Doudieu, V. Stümpflen, and H. W. Mewes. Corum: the comprehensive resource of mammalian protein complexes. *Nucleic Acids Research*, 36(Database issue):D646–D650, 2008.

[26] J. Scott, T. Ideker, R. M. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13(2):133–144, March 2006.

[27] SGD project. Saccharomyces genome database. http://www.yeastgenome.org/, Sept. 2008.

[28] R. Sharan, T. Ideker, B. P. Kelley, R. Shamir, and R. M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Journal of Computational Biology*, 12(6):835–846, 2005.

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

[30] F. Sohler and R. Zimmer. Identifying active transcription factors and kinases from expression data using pathway queries. *Bioinformatics*, 21 (Suppl. 2):ii115–ii122, 2005.

[31] 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.

[32] U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F. H. Brembeck, H. Goehler, M. Stroedicke, M. Zenkner, A. Schoenherr, S. Koeppen, et al. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968, 2005.

[33] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[34] I. Xenarios, L. Salwnski, J. X, P. Higney, K. S, and E. D. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30:303–5, 2002.

[35] Q. Yang and S.-H. Sze. Path matching and graph matching in biological networks. *Journal of Computational Biology*, 14(1):56–67, 2007.

[36] Yu et al. High-quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110, 2008.

[37] Y. Zheng, J. D. Szustakowski, L. Fortnow, R. J. Roberts, and S. Kasif. Computational identification of operons in microbial genomes. *Genome Research*, 12(8):1221–1230, 2002.