JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 17, Number 3, 2010

Pp. 237–252
DOI: 10.1089/cmb.2009.0170

Topology-Free Querying of Protein Interaction Networks

SHARON BRUCKNER,[1] FALK HÜFFNER,[1] RICHARD M. KARP,[2]
*RON SHAMIR,[1] and *RODED SHARAN[1]

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 in terms of sequence, topology, or both. 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 address 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 giving results that are highly functionally coherent.

Key words: algorithms, gene expression, gene networks, genetic variation, sequence analysis.

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.

In its simplest form, when ignoring sequence similarity, and requiring exact match of edges in the query and target subnetworks, this problem corresponds to the NP-hard SUBGRAPH ISOMORPHISM problem. Typically, further restrictions are imposed on target nodes a query node can map to, for example, based on sequence similarity of the corresponding proteins in a protein–protein interaction (PPI) network.

The largest body of previous work on network querying concerns querying subnetworks across species. Kelley et al. (2004) and later Shlomi et al. (2006) devised fixed-parameter algorithms for querying linear paths within a PPI network. These algorithms were subsequently extended in the QNet software to allow

[1]School of Computer Science, Tel Aviv University, Tel Aviv, Israel.
[2]International Computer Science Institute, Berkeley, California.
*These authors contributed equally to this work.

searching for trees and bounded treewidth graphs (Sharan et al., 2008). A related work by Pinter et al. (2005) presented a polynomial algorithm for detecting homeomorphic subtrees within a tree (representing a collection of metabolic pathways). Another polynomial approach that relaxes the homomorphism but requires target and query nodes to agree in their neighborhoods was given by Narayanan and Karp (2007). Sohler and Zimmer (2005) 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 (2007) examined both path queries and general ones, but since their method is based on exhaustive enumeration, it can handle small queries only. A different approach to network querying is to apply *network alignment* (Kelley et al., 2003) algorithms. Network alignment was introduced by Ogata et al. (2000) with the aim of detecting functionally related enzyme clusters (FRECs) by aligning metabolic networks. Another work by Tohsato et al. (2000) is concerned with the comparative analysis of metabolic pathways and allows querying linear paths. In general, network alignment algorithms can be adapted to the network querying problem by aligning a query and a network instead of two networks. See e.g. Kelley et al. (2003) for more information.

Another line of work on network querying, and one that is similar to our own, has been the search for small motifs defined in terms of the functional attributes of their member proteins, and—in some cases—the interactions among them. Lacroix et al. (2006) 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. The method was implemented in a program called MOTUS (Lacroix, 2009). NetGrep (Banks et al., 2008) is a web-tool for identifying subgraphs of interest where the algorithm is based on heuristic pruning of the possible solution space. A major advantage of this method is its running time, which is fast due to the emphasis on implementation and the highly specialized nature of their small queries. Betzler et al. (2008) gave a fixed-parameter algorithm for the latter problem and some extensions of it. An additional heuristic solution was offered by Zheng et al. (2002) to a similar problem, in the context of querying metabolic networks. Finally, Ferro et al. (2008) presented the GraphFind algorithm, which utilizes fast heuristics for subgraph isomorphism to identify approximate matches of queries within a collection of networks. The Cytoscape (Shannon et al., 2003) plugin NetMatch (Ferro et al., 2007) implements the ideas of GraphFind.

A limitation of the approaches above—except for Betzler et al. (2008) and Lacroix et al. (2006)—is that they rely on precise information on the interaction pattern of the query pathway. However, this information is often missing. For example, hundreds of protein complexes have been reported in the literature for yeast (SGD Project, 2008), human (Ruepp et al., 2008), and other species, but for most of these complexes no information exists on their interaction patterns (Yu et al., 2008), 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 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 dynamic programming (DP) algorithms that utilize the color-coding paradigm (Alon et al., 1995) for several variants of this problem. In addition, we provide an integer linear programming (ILP) formulation of it. This 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. We also develop a fast heuristic approach to the problem. By using a combination of the three 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 (Sharan et al., 2008) topology-based approach, and on queries from less studied species, where only topology-free algorithms apply, where we also compared to the methods of Lacroix et al. (2006) implemented in the MOTUS software. TORQUE detected many more matches than QNet and MOTUS, while in both cases giving results that are highly functionally coherent.

A preliminary version of this study appeared in the proceedings of the 13th RECOMB conference (Bruckner et al., 2009a). A paper describing the TORQUE web-server implementing these algorithms appeared in Bruckner et al. (2009b).

2. PRELIMINARIES

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 $\Gamma : V \to 2^C$ associates with each $v \in V$ a subset of colors $\Gamma(v) \subseteq C$. 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 $\Gamma(v)$, 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 $\Gamma : V \to 2^C$, is there a connected subgraph of G that is C-colorful?*

This problem was shown to be NP-complete by Fellows et al. (2007), even for the case of trees of maximum degree 3. Here we provide fixed-parameter 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 $f(k) \cdot n^{O(1)}$ time, where f is an arbitrary function. Thus, fixed-parameter algorithms allow solving relatively large instances of NP-hard problems exactly (Niedermeier, 2006), as long as the parameter value is modest.

We present three different approaches to Problem 1: A dynamic programming-based randomized algorithm is presented in Section 3, an integer linear programming formulation is described in Section 4, and in Section 5 we present a fast heuristic approach. Later sections describe implementation, experimental results, and conclusions.

3. DYNAMIC PROGRAMMING

In this section, we show how to solve Problem 1 using a randomized dynamic programming approach. We first 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. In Section 3.3, we give a reduction from the general case to the single color case.

Since every connected subgraph has a spanning tree, it suffices to look for colorful trees. This problem has been studied by Scott et al. (2006) in another context, as well as by Kalaev et al. (2008) and Betzler et al. (2008). For completeness, we provide a dynamic programming (DP) algorithm, which is the unweighted version of the algorithm given by Scott et al. (2006). 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, 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.[1] One can easily generalize (1) to the weighted case, where each edge is assigned a weight, and the heaviest tree is sought. In this case, the algorithm finds, for each vertex, the heaviest (maximum total edge weight) colorful subtree rooted at it. We note that this does not guarantee finding the heaviest subgraph containing a given vertex. The latter variant seems not to be amenable to this approach, unless some structure is assumed on the heaviest subgraph (such as being of bounded treewidth).

Initialization: For $S=\{\gamma\}$ define $B(v,\gamma)=0$ iff $\Gamma(v)=\gamma$, $B(v,\gamma)=-\infty$ otherwise.

[1]It can be further reduced to $O(2^k m)$ using the techniques of Björklund et al. (2007); however, this version cannot be generalized to the weighted case, so we do not use it.

Recursion:

$$B(v,S) = \max_{\substack{u \in N(v) \\ S_1 \uplus S_2 = S \\ \Gamma(v) \in S_1, \Gamma(u) \in S_2}} B(v,S_1) + B(u,S_2) + w(u,v), \tag{2}$$

where $B(v,S)$ is now a real number instead of a Boolean value. The weight of an optimum match is given by $\max_v B(v,C)$.

3.1. 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_{del} deletions can be done by scanning the entries of B. If there exists $\hat{C} \subseteq C$ such that $|\hat{C}| \geq |C| - N_{\text{del}}$ and $B(v,\hat{C}) = True$, then a valid solution exists.

When allowing insertions, there are several problem variants to consider (Fig. 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*.

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. (1995): 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

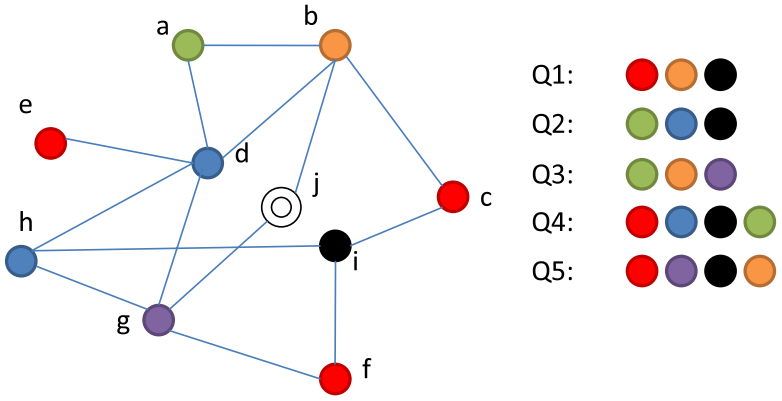

FIG. 1. Network query problems. (**Left**) The network, where vertex j is non-colored. (**Right**) Queries. For the basic problem disallowing indels, $Q1$ is solved by $\{c,b,i\}$, while $Q2$ and $Q4$ have no solution. When allowing a single arbitrary insertion, $Q2$ has solution $\{a,d,h,i\}$ and $Q4$ 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 $\{b,c,i,f,g\}$.

initialize the table by setting all entries to *False*, except: (i) For $\gamma \neq 0$, $B(v, \{\gamma\}, 0)$ iff $\Gamma(v) = \gamma$; and (ii) if $\Gamma(v) = 0$, $B(v, \emptyset, 1)$. Entries for which $|S| \geq 1$ and $j > 0$ are then computed using the following recurrence:

$$B(v,S,j) = \Big[\bigvee_{\substack{u \in N(v) \\ S_1 \uplus S_2 = S \\ j_1 + j_2 = j}} B(v,S_1,j_1) \wedge B(u,S_2,j_2) \Big] \wedge \forall j' < j : \neg B(v,S,j'). \tag{3}$$

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:

$$\begin{aligned} B(v,\{\gamma\},j) \Longleftrightarrow \exists u \in N(v), S_1, S_2, j_1, j_2 : \\ B(v,S_1,j_1),\ B(u,S_2,j_2),\ S_1 \uplus S_2 = \{\gamma\},\ j_1 + j_2 = j,\ \forall j' < j : \neg B(v,\{\gamma\},j'). \end{aligned} \tag{4}$$

We prove the case $|S| = 1, j > 0$ by induction over j. 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 $\Gamma(v) = 0$, $j_1 = 1$, 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 induction over $|S|$ and j. By definition:

$$\begin{aligned} B(v,S,j) \Longleftrightarrow \exists u \in N(v), S_1, S_2, j_1, j_2 : \\ B(v,S_1,j_1),\ B(u,S_2,j_2),\ S_1 \uplus S_2 = S,\ j_1 + j_2 = j,\ \forall j' < j : \neg B(v,S,j'). \end{aligned} \tag{5}$$

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 evaluated to j when for the first time $B(v, S, j)$ is true for some j. In the recursion (3), we replace the condition $j_1 + j_2 = j$ by $t(v, S_1) + t(u, S_2) = j$. Since the table is $(N_{ins} + 1)$ 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. ■

3.2. Arbitrary insertions

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 $E' = E \cup \{(v^0, u) \mid (v, u) \in E\} \cup \{(v^0, u^0) \mid (v, u) \in E\}$. Asymptotically, this does not change the running time.

Theorem 2. *The above algorithm solves the arbitrary insertions variant in time* $O(N_{ins} 3^k m)$.

Proof. A match in G can be translated to a match in G' by replacing vertices of repeating colors with their copies. Conversely, let $\tilde{T} \subseteq G'$ be a subgraph satisfying the coloring constraints, having j special insertions. Let $U = \tilde{T} \cap V$ and $U' = \tilde{T} \setminus U$. The crucial observation is that $\tilde{T}$ does not contain a pair $\{v, v^0\}$, as otherwise we obtain a contradiction to the minimality of j (we can replace each edge (u, v^0) with an edge (u, v) and obtain a solution having $j - 1$ insertions). It follows that $U \cup \{v \mid v^0 \in U'\}$ is a valid match in G.

The bound on the running time follows from the observation that the size of G' is linear in the size of G. ■

3.3. 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. (2008) 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 $(v, c) \in E \iff c \in \Gamma(v)$. 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 3. *The probability P_c for a subset of vertices of size k to become colorful in a random coloring is at least $\frac{1}{k!}$.*

Proof. Take a solution set $S = \{v_1, \ldots, v_k\}$ and the color subgraph B' induced by $S \cup C$. Let d_i denote the degree of v_i in B'. Suppose, w.l.o.g., that $d_1 \geq \ldots \geq d_k$. Finally, denote by $\mathrm{Perm}(B')$ the number of perfect matchings in B'. Note that there is a 1-1 mapping between colorful colorings of S in G and perfect matchings in B'.

The probability in question is equal to the ratio of number of perfect matchings to the number of ways to color the vertices of the solution, i. e., $P_c = \frac{\mathrm{Perm}(B')}{\prod_{i=1}^{k} d_i}$. We claim that if $\mathrm{Perm} > 0$ then $P_c \geq \frac{1}{k!}$. Indeed, under the conditions set above, Ostrand (1970) proved the following bound:

$$\mathrm{Perm}(B') \geq \prod_{i=1}^{k} \max\{1, d_i - i + 1\} \tag{6}$$

Let $D_1 = \{d_i \mid d_i \geq i\}$ and $D_2 = \{d_i \mid d_i < i\}$. Observe that if $d_i \in D_1$, then $\frac{d_i - i + 1}{d_i} \geq \frac{1}{i}$. Otherwise, $d_i < i$ and $\frac{1}{d_i} > \frac{1}{i}$. Thus,

$$P_c \geq \frac{\prod_{i=1}^{k} \max\{1, d_i - i + 1\}}{\prod_{i=1}^{k} d_i} = \prod_{i \in D_1} \frac{d_i - i + 1}{d_i} \prod_{i \in D_2} \frac{1}{d_i} \geq \prod_{i \in D_1} \frac{1}{i} \prod_{i \in D_2} \frac{1}{i} = \prod_{i=1}^{k} \frac{1}{i} = \frac{1}{k!}. \tag{7}$$

■

Theorem 3 implies an overall running time of $O(k!3^k m N_{\mathrm{ins}}^2)$ in the case of multiple color constraints. However, this bound is excessive in many instances, for the following reason. Let V' be a set of colored vertices. Following Sharan et al. (2008), 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, the expected number of iterations of the algorithm is also bounded by $\prod_{i=1}^{s} |P_i|!$.

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 (Lovász and Plummer, 1986), 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.

In practice, we find that the above reductions bring the number of required iterations to under 100 in the majority of cases. This is considerably less than the theoretical bound proposed by Betzler et al. (2008). For example, when querying for human complexes in yeast, we obtain an improvement of at least 50% in the number of iterations in 45% of the complexes.

We apply a preprocessing step in each iteration of the DP algorithm to reduce the graph size. Each such iteration assigns each vertex a single color from its set of allowed colors. We then look at the subgraph H induced by the set of vertices that are assigned a *private color*, i.e., a color that was not assigned to any

other vertex. We split H into its connected components, and merge the vertices of each connected component to a new vertex. Each new vertex is assigned a new color, and this reduced set of new colors replaces the private colors. The DP algorithm then receives as input a reduced graph and set of colors, resulting in a faster running time and no effect on the quality of the results.

4. INTEGER PROGRAMMING FORMULATION

In this section, we provide an ILP formulation of the network querying problem, allowing us to employ ILP 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 and exactly N_{del} arbitrary deletions. Further, we are given edge weights $\omega : E \to \mathbb{Q}$ and wish to find a vertex subset $K \subseteq V$ of size $t := k + N_{\text{ins}} - N_{\text{del}}$ 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[2] 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:

$$\{c_v : v \in V\}, c_v \in \{0,1\} \quad \text{vertex } v \text{ is selected } (v \in K) \tag{8}$$

$$\{e_{vw} : (v,w) \in E, v < w\}, e_{vw} \in \{0,1\} \quad \text{edge } (v,w) \text{ is in } G[K] \tag{9}$$

$$\{r_v : v \in V\}, r_v \in \{0,1\} \quad \text{vertex } v \text{ is the sink} \tag{10}$$

$$\{f_{vw}, f_{wv} : (v,w) \in E\}, f_{vw}, f_{wv} \in \mathbb{Q} \quad \text{flow from } v \text{ to } w/w \text{ to } v \tag{11}$$

$$\{g_{v\gamma} : v \in V, \gamma \in \Gamma(v)\}, g_{v\gamma} \in \{0,1\} \quad \text{vertex } v \text{ has color } \gamma \tag{12}$$

and the following constraints

$$\sum_{v\in V} c_v = t \tag{13}$$

$$\sum_{v\in V} r_v = 1 \tag{14}$$

$$e_{vw} \le c_v \wedge e_{vw} \le c_w \quad \forall (v,w) \in E \tag{15}$$

$$2e_{vw} \ge c_v + c_w - 1 \quad \forall (v,w) \in E \tag{16}$$

$$f_{vw} = -f_{wv} \quad \forall (v,w) \in E \tag{17}$$

$$\sum_{w\in N(v)} f_{vw} = c_v - t r_v \quad \forall v \in V \tag{18}$$

$$f_{vw}, f_{wv} \le (t-1) e_{vw} \quad \forall (v,w) \in E \tag{19}$$

$$\sum_{\gamma\in\Gamma(v)} g_{v\gamma} \le 1 \quad \forall v \in V \tag{20}$$

$$\sum_{v\in V} g_{v\gamma} \le 1 \quad \forall \gamma \in C \tag{21}$$

$$\sum_{v\in V} \sum_{\gamma\in\Gamma(v)} g_{v\gamma} = t - N_{\text{ins}} \tag{22}$$

$$g_{v\gamma} \le c_v \quad \forall v \in V, \gamma \in \Gamma(v) \tag{23}$$

[2]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; for an introduction on flows, see Cormen et al. (2001).

with the objective

$$\text{maximize} \sum_{(v,w)\in E} \omega_{vw} e_{vw}. \tag{24}$$

We now turn to proving the correctness of the formulation above.

Theorem 4. *The ILP defined by* (8)–(24) *correctly solves Problem 1.*

Proof. The integrality constraints (8) and (9) and the inequalities (15) and (16) ensure that $e_{vw} = 1$ if and only if $c_v = 1 \wedge c_w = 1$. Therefore, the two objectives match. It remains to show that the constraints of Problem 1 are met if and only if the constraints (13)–(23) are met.

"⇒": Given a solution of Problem 1, set c_v, e_{vw}, and $g_{v\gamma}$ as described in (8), (9), and (12), respectively. Select an arbitrary $r \in K$ and set $r_r = 1$ and $r_v = 0$ for each $v \in V, v \neq r$. Let T be a spanning tree of $G[K]$ (which exists because $G[K]$ is connected) with edges directed towards r. Set f_{vw} for $(v,w) \in E(T)$ to the number of vertices in the subtree of T rooted in v, and set $f_{wv} = -f_{vw}$ and $f_{vw} = 0$ for $v \notin K$ or $w \notin K$. It is then easy to verify that (13)–(23) hold.

"⇐": Given a solution of the ILP, let K, r, and c be defined by (8), (10), and (12), respectively. Because of (13), we have $|K| = \mathrm{t}$, and because of (14), r is well-defined. Constraints (20) make g be well-defined, and (21) make sure it is colorful. Because of (22), there are exactly $t - N_{\text{ins}}$ colored vertices, and with (23) they must all be in K, implying that there are exactly N_{ins} uncolored vertices in K.

It remains to show that $G[K]$ is connected. For this, we show that every $v \neq \mathrm{r} \in K$ has a path to r consisting only of vertices from K. Constraint (17) ensures flow skew symmetry. Constraint (18) ensures that the outgoing flow $\sum_{w\in N(v)} f_{vw}$ is 1 for $v \in K$, $v \neq r$ and 0 for $v \notin K$. For the root r, we always have $c_r = 1$, since (17) and (18) requires at least one edge with positive flow incident on r, and so (19) and (15) force $c_r = 1$. Thus, for the root r, the outgoing flow is $-(t-1)$, and f forms a valid flow with $(t-1)$ sources $K \setminus \{r\}$ and one sink r.

Consider now $v \neq \mathrm{r} \in K$. Because of (18), v has at least one neighbor w with $f_{vw} \geq 1$. Because of (19), we have $w \in K$. We continue this way until reaching r or reaching a vertex w' for the second time. If we reached w' again, we can decrease the flow on all edges traversed after w' by 1, yielding another valid flow without violating any of (17), (18), and (19). This will eventually provide a path to r.

5. SHORTEST-PATH BASED HEURISTIC

The similarity measure we employ between complex and network proteins is homology-based. We often observed that the majority of network proteins were not sufficiently similar to any complex protein, and can be used only as "special" insertion vertices. Therefore, in practice, only a small fraction of the network proteins was assigned any color. We thus developed a fast heuristic that solves problem 1 when the number of colored vertices is small, without allowing indels. We then use this method as a preliminary step, accepting the solutions it returns and running the DP or ILP algorithm in the cases where the heuristic returns no solution (either because it failed to find one, or because indels are required).

Our heuristic is based on a shortest-path algorithm to obtain a fast solution. Several fast iterations are run. During each iteration, one new vertex is added to the solution while a set of vertices is removed from the network, making the problem smaller. The model includes weighted edges and supports multiple colors per vertex. A "vertex of color c" in this context is a vertex that has c in its coloring constraints.

The algorithm maintains a partition of V into three sets, V_{in}, V_{out}, and V_{open} Starting with $V_{\text{open}} = V$, vertices are greedily moved from V_{open} either to V_{in}, meaning that they are to be part of the solution set H, or to V_{out}, meaning that they are not to be part of H.

We define several dynamically changing variables, functions and properties determined by the partition $(V_{\text{in}}, V_{\text{out}}, V_{\text{open}})$.

- A vertex in V_{open} is *unique* if there is no other vertex of the same color in V_{open}.
- For v in V_{open}, let $d(v)$ denote the number of edges in a shortest path in $G[V_{\text{in}} \cup V_{\text{open}}]$ from v to the closest vertex of V_{in} (or ∞ if no such path exists.)

- For any color c such that V_{open} contains at least one vertex of color c, let $\text{dist}(c) = \min_{v|c\in\Gamma(v)} d(v)$. Let $c^* = \text{argmax}_c\ \text{dist}(c)$. Let v^* be a vertex with color c^* and $d(v^*) = \text{dist}(c^*)$. If there are several options for v^*, break ties according to the sum of the edge weights along the path from the candidate to V_{in}.

The following actions occur automatically at any point in the algorithm and take priority over the other steps of the algorithm.

- If a vertex becomes unique it is moved from V_{open} to V_{in};
- If a vertex is placed in V_{in}, all other vertices of the same color are placed in V_{out}, if they cannot be assigned any other colors that are not already represented in V_{in}.
- If $v \in V_{\text{open}}$ and $d(v) = \infty$ then v is placed in V_{out}.

We assume that there is at least one unique vertex at the beginning of the algorithm. If there are no unique vertices, we choose the least frequent color and run the heuristic several times, keeping only a single vertex of this color and removing the rest each time. A connected component of $G[V_{\text{in}} \cup V_{\text{open}}]$ is called *essential* if there is some color c such that the component contains a vertex of color c, and no other component contains a vertex of color c. The partition $(V_{\text{in}}, V_{\text{out}}, V_{\text{open}})$ is called *bad* if $G[V_{\text{in}} \cup V_{\text{open}}]$ contains two or more essential components. The algorithm tries to keep adding vertices to V_{in} without creating a bad partition. A path in G is called *feasible* if it can be colored in such a way that every vertex on it has a different color.

Now we are ready to describe a general step of the algorithm, in which V_{in}, V_{out}, and V_{open} are given. We may assume that there are no unique vertices in V_{open}, since they would have been automatically added to V_{in}.

General step: Execute the first case for which the precondition holds.
Case 1. $G[V_{\text{in}}]$ is connected and V_{open} is empty: Return SUCCESS
Case 2. $G[V_{\text{in}}]$ is connected or the degree $d(v^*)$ in the graph induced by $V_{\text{in}} \cup V_{\text{open}}$ is ≥ 3: $V_{\text{in}} \leftarrow V_{\text{in}} \cup \{v^*\}$.
Case 3. There is no feasible path joining two connected components of $G[V_{\text{in}}]$: Return FAILURE

Choose a shortest feasible path joining two connected components of $G[V_{\text{in}}]$. This path can be found, for example, by a simple depth search first (DFS) procedure, where we also keep a list of all possible color combinations that are feasible along the path. When a path is extended and it has several options of colors, combinations repeating a color are eliminated. If there are several options for this path, break ties according to the sum of the edge weights along it. Let the first vertex of V_{open} in that path be v.

Case 4. Adding v to V_{in} and the other vertices of the same color as v to V_{out} does not create a bad partition: $V_{\text{in}} \leftarrow V_{\text{in}} \cup \{v\}$
Case 5. There is a vertex w of the same color as v such that adding w to V_{in} does not create a bad partition: $V_{\text{in}} \leftarrow V_{\text{in}} \cup \{w\}$
Case 6. Return FAILURE

6. IMPLEMENTATION

We implemented a pipeline called TORQUE for querying a complex given as a set of proteins from a source species in the PPI network of a target species. TORQUE runs with increasing number of allowed indels until a match is found or a pre-specified bound on the number of indels is reached. Matches are assigned a score based on edge weights, and the highest scoring match is finally output. The problem version addressed is the multiple colors per vertex model with arbitrary insertions. Before applying the computationally intensive DP or ILP methods, we try the fast heuristic based on shortest paths that does not allow indels but works well in practice on small instances (our tests show it returns a good match about 60% of the time). We 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, 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. A component is *feasible* if the color constraints of its vertices contain at least $k - N_{\text{del}}$ colors of the query. Next, we process feasible connected components of increasing size, searching for the highest scoring matched complex using any of our 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 methods will find a match more efficiently. In the case when the connected component does not contain any special insertion vertices, we begin with the shortest-path heuristic, which terminates quickly. If it returns a match, we accept it as the solution for this connected component. Otherwise, we attempt either the DP or the ILP methods. While the ILP solution is faster on many cases, its running time is unpredictable, unlike the DP algorithm, which tends to be more stable in running time and also guarantees optimality or determines that no solution exists. We therefore chose to use it only in certain cases: 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 observed that its running time is less sensitive to k, while the color-coding algorithm is exponential in k. Based on empirical tests, 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 connected components we tested. For the DP algorithm, we used the multiple colors per vertex model, generating color assignments for the vertices using the bounds described in Section 3.3, thus reducing the number of iterations required.

Scoring. We score a set of proteins matching a query using the approach of Sharan et al. (2005). 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) versus the chance that its connections in the target network arise at random. The protein complex model assumes that every two proteins in a complex should interact, independently of all other pairs, with high probability β. The random model assumes that the PPI graph was chosen uniformly at random from the collection of all graphs with the same vertex degrees as the observed one. This random model induces a probability of occurrence p_{uv} for each edge (u, v) of the graph. 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. Overall, for a match U, the likelihood ratio score is expressed as a sum over the vertex pairs in the match:

$$\mathcal{L}(V) = \sum_{(u,v)\in U\times U} \log \frac{\beta Pr(O_{uv} \mid T_{uv}) + (1-\beta) Pr(O_{uv} \mid F_{uv})}{p_{uv} Pr(O_{uv} \mid T_{uv}) + (1-p_{uv}) Pr(O_{uv} \mid F_{uv})}, \tag{25}$$

where O_{uv} denotes the set of experimental observations on the interaction status of u and v, T_{uv} denotes the event that u and v truly interact, and F_{uv} denotes the event the u and v do not interact. The computation of $Pr(O_{uv} \mid T_{uv})$ and $Pr(O_{uv} \mid F_{uv})$ is based on the reliability assigned to the interaction between u and v.

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.

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}. Such protein pairs are *sequence similar*. For

each complex, we allowed TORQUE to run at most one hour, and took the best solution up to that point. We set the maximum 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.

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

7. 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 (Sharan et al., 2008).[3] 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. We compared our results for all data collections in yeast with those of MOTUS (http://genome.imim.es/~vlacroix/motus/), a program for querying for colorful motifs in networks that is based on the work of Lacroix et al. (2006). In the following, we describe the data, the evaluation measures, and the results obtained.

Data acquisition. For yeast, fly, and human, we used the networks recently published by Yosef et al. (2009). Their networks were obtained using up-to-date PPI data gathered from several papers (Stanyon et al., 2004; Rual et al., 2005; Stelzl et al., 2005; Gavin et al., 2006; Krogan et al., 2006; Reguly et al., 2006) and from public databases (Xenarios et al., 2002; FlyBase-Consortium, 2003; Peri et al., 2003). High-throughput mass spectrometry data (Gavin et al., 2006; Krogan et al., 2006) was translated into binary PPIs using the spoke model (Bader and Hogue, 2002). Yeast complexes were downloaded from SGD (SGD project, 2008) (Macromolecular Complex GO-Slim category). Fly complexes were obtained using the AmiGo (GO Consortium, 2008) 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 (Ruepp et al., 2008).

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 Ontology (GO) (GO Consortium, 2000) 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 (Boyle et al., 2004) tool. The p-values returned by the tool were further corrected for multiple match testing using the false discovery rate (FDR) procedure (Benjamini and Hochberg, 1995).

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. This specificity computation was applied to the 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 (Sharan et al., 2008), 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

[3]A comparison to GraphFind (Ferro et al., 2008) was not feasible, since its interface does not allow automated execution of the more than 600 queries.

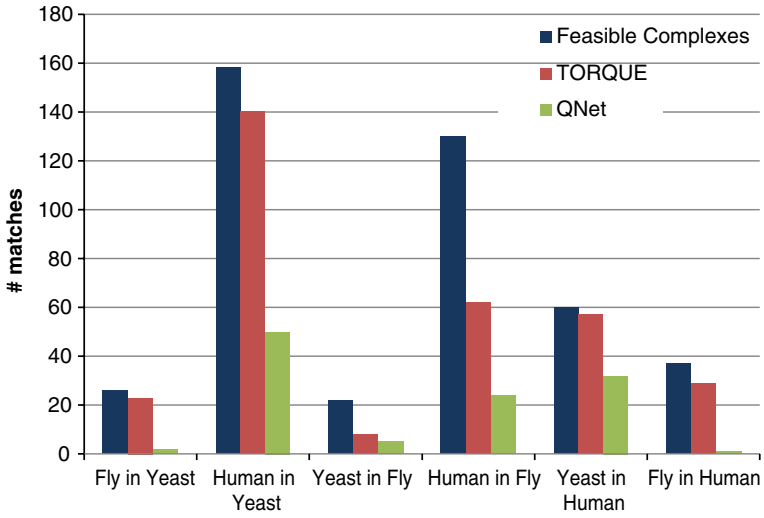

FIG. 2. Comparison of number of matches for TORQUE and QNet.

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 of them, while QNet found matches for 114 only. As we show below, this 170% 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. As our functional coherence results are highly significant, it may be possible to predict new annotations for the proteins in our solutions that are not enriched with the GO terms known for the query complex.

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 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.

Comparison to MOTUS. MOTUS (Lacroix, 2009) is software for colored motif search and inference in vertex colored graphs. Its inputs are a network (list of unweighted edges), where each vertex has a label,

TABLE 1. QUALITY RESULTS

		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

The table lists the number and percentage of matches, out of all found matches, that pass a significance threshold of 0.05, and the number of novel complexes detected.

TABLE 2. STATISTICS OF QUERYING PROTEIN COMPLEXES FOR WHICH NO TOPOLOGY INFORMATION IS AVAILABLE

Network	*Complex*	*No. feasible*	*No. 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		191	162	122	83	37

and a query motif, modeled as a set of labels. MOTUS then lists all connected occurrences of this motif in the network. The version we tested, provided by the authors in March 2009, does not include support for multiple labels per vertex (equivalent to our multiple colors per vertex), deletions and gaps (insertions), or weighted edges. Therefore, we tested MOTUS on our data in the cases where every network protein was similar to at most one query protein, implying that there is only a single color (or label) per vertex. We removed from the query all nodes that had no similar vertex in the network at all (a priori deletions). For each occurrence found, we looked at the subgraph induced by this occurrence on our original weighted-edges network, and scored the subgraph as described in Section 6. We tested MOTUS on complexes from all species queried in the yeast PPI network. On this reduced set of instances, MOTUS found a match for 44 out of 220 feasible complexes ($\sim$20%), while TORQUE found a match in 199 ($\sim$90%) of the cases. MOTUS returned a match for most small motifs (sizes 3–5) that did not require insertions or deletions, while larger motifs or those that TORQUE matched using indels were usually not found within the time constraint of one hour and the memory constraint of 8 GB. Since MOTUS finds all possible occurrences of the motif, the optimal, highest scoring solution can also be found in its output.

Indels and running time. A major advantage of our approach is its flexibility in allowing insertions and deletions. Indeed, only 80 of our 482 identified matches required no insertions or deletions at all. The number of allowed insertions affects the size of the components tested, and therefore affects the running time. In general, 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 tested. Figure 3 gives the running time

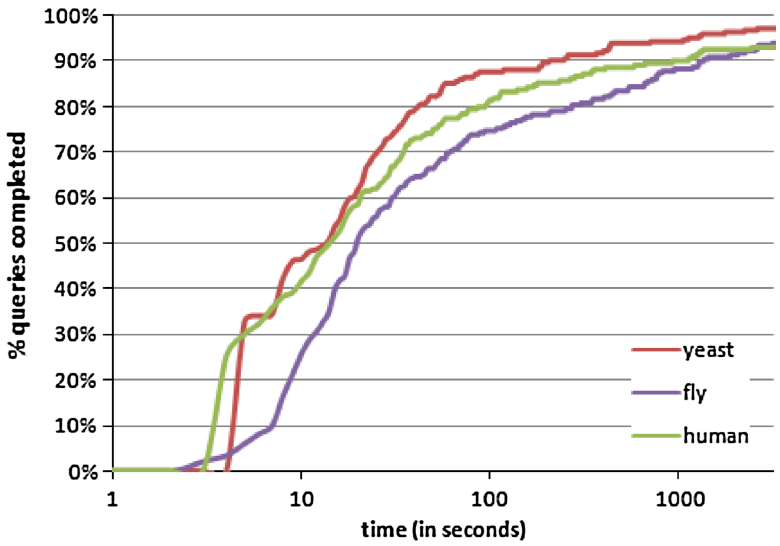

FIG. 3. Runtime distribution of queries. The figure shows the cumulative percentage of queries completed within each running time. Queries were stopped after 3600 seconds, so 100% is not reached.

distribution of all queries, i.e., the fraction that were completed within each time frame, for each of the three target networks. As the graph shows, more than 30% of queries from all networks were processed in 10 seconds or less, and 100 seconds are enough for more than 70% of queries. For a few of the queries, the running time required is higher; however, approximately 95% of the queries ended under the 1 hour time bound. Of the 624 feasible complexes, only 32 did not end in time. The three networks show similar behavior, and we attribute the minor differences among them to the difference in the quality and completeness of the networks.

8. CONCLUSION

In this study, we presented a method and a tool for querying protein complexes within a PPI network, where no topology information about the complex is available. Our method combines three algorithms: a dynamic programming exact algorithm, an integer linear programming formulation, and a fast heuristic. Compared to a topology-based approach and to heuristics for colorful motif finding, our approach produces substantially more matches, while preserving high functional coherence. Thus, our tool is appropriate for a wide range of network query tasks, and in particular when interaction data on the query species are sparse or unavailable. Our method also suggests new complexes for which no prior experimental evidence is available, as they do not overlap any known complex. Checking these hypotheses experimentally is an interesting next step.

Our approach has several limitations that we hope to address in future work:

- The running time for some queries is still prohibitively high, depending on the problem size and the number of colors (query size) and their distribution. We are examining additional heuristics in an attempt to bring the running time down further.
- The integer linear programming formulation proved very powerful, and usually performs well in practice, especially when commercial software was used. However, the performance depends on the availability of such software, and the specific algorithm chosen by such software is often not transparent and hence its behavior is less predictable.
- Since our methods do not rely on the topology of the query, they do not depend on PPI data from the query species. However, our approach is sensitive to the quality of the PPI data in the target species, and specifically to the false negative rate in that network (fraction of true interactions that are still unknown). A true complex in the target network, which biologically is a match to the query complex, may not be found if the corresponding subnetwork is disconnected due to false negative edges. This can be seen, for example, in the fly network, which is noisier than the human and mouse networks: there are many feasible queries for which TORQUE finds no solution. False positive edges in the target network are less problematic, as they do not disrupt the connectivity (but can create spurious solutions).
- Extending the notion of similarity between query and target proteins might also improve the solutions. Currently we rely on sequence similarity, and a fixed threshold for sequence similarity is used. Combining other similarity measures such as structural similarity could be considered. Furthermore, replacing the threshold by a weighted model could improve the results.

On the theoretical side, several follow-up directions are of interest. It would be interesting to see how the ideas from the ILP concerning modeling of connected subgraphs can be adapted to other network problems such as identification of significantly responding subnetworks (''activity modules'') (Ulitsky and Shamir, 2007). The main problem that we addressed can be generalized also to allowing special insertions where colored vertices may repeat freely without penalty for reuse. The DP approach can be generalized to cover that version as well, and we intend to implement and test it.

Finally, a web-server that allows the use of Torque for online queries has recently been implemented and is available at www.cs.tau.ac.il/~bnet/torque.html. For further details, see Bruckner et al. (2009b).

ACKNOWLEDGMENTS

We thank Noga Alon for help in analyzing the case of multiple color constraints. We thank Banu Dost for the QNet code, Vincent Lacroix for the MOTUS code, and Nir Yosef for 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.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Alon, N., Yuster, R., and Zwick, U. 1995. Color coding. *J. ACM* 42, 844–856.

Bader, G.D., and Hogue, C.W. 2002. Analyzing yeast protein-protein interaction data obtained from different sources. *Nat. Biotechnol.* 20, 991–997.

Banks, E., Nabieva, E., Peterson, R., and Singh, M. 2008. NetGrep: fast network schema searches in interactomes. *Gen. Biol*. 9.

Benjamini, Y., and Hochberg, Y. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc. Ser. B* 57, 289–300.

Betzler, N., Fellows, M.R., Komusiewicz, C., et al. 2008. Parameterized algorithms and hardness results for some graph motif problems. *Lect. Notes Comput. Sci.* 5029, 31–43.

Björklund, A., Husfeldt, T., Kaski, P., et al. 2007. Fourier meets Möbius: fast subset convolution. *Proc. 39th STOC* 67–74.

Boyle, E.I., Weng, S., Gollub, J., et al. 2004. 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, 3710–3715.

Bruckner, S., Hüffner, F., Karp, R.M., et al. 2009a. Topology-free querying of protein interaction networks. *Lect. Notes Bioinform.* 5541, 74–89.

Bruckner, S., Hüffner, F., Karp, R.M., et al. 2009b. Torque: topology-free querying of protein interaction networks. *Nucleic Acids Res.* 37, W106–W108.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., et al. 2001. *Introduction to Algorithms*, 2nd ed. MIT Press, Cambridge, MA.

Fellows, M.R., Fertin, G., Hermelin, D., et al. 2007. Borderlines for finding connected motifs in vertex-colored graphs. *Lect. Notes Comput. Sci.* 4596, 340–351.

Ferro, A., Giugno, R., Mongiovì, M., et al. 2008. GraphFind: enhancing graph searching by low support data mining techniques. *BMC Bioinform.* 9, Suppl 4, 1471–2105.

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, 910–912.

FlyBase-Consortium. 2003. The FlyBase database of the Drosophila genome projects and community literature. *Nucleic Acids Res.* 31, 172–175.

Gavin, A.C., Aloy, P., Grandi, P., et al. 2006. Proteome survey reveals modularity of the yeast cell machinery. *Nature* 440, 631–636.

Go Consortium (The Gene Ontology Consortium). 2000. Gene ontology: tool for the unification of biology. *Nat. Genet.* 25, 25–29.

GO Consortium. 2008. Amigo. Available at: http://amigo.geneontology.org/. Accessed December 1, 2009.

Kalaev, M., Bafna, V., and Sharan. R. 2008. Fast and accurate alignment of multiple protein networks. *Lect. Notes Comput. Sci.* 4955, 246–256.

Kelley, B.P., Sharan, R., Karp, R.M., Sittler, T., Root, D.E., Stockwell, B.R., and Ideker, T. 2003. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc. Nat. Acad. Sci.* 100, 11394–11399.

Kelley, B.P., Yuan, B., Lewitter, F., et al. 2004. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Res.* 32.

Krogan, N.J., Cagney, G., Yu, H., et al. 2006. Global landscape of protein complexes in the yeast *Saccharomyces cerevisiae*. *Nature* 440, 637–643.

Lacroix. V. 2009. Motus. Available at: http://genome.imim.es/∼vlacroix/motus/. Accessed December 1, 2009.

Lacroix, V., Fernandes, C., and Sagot, M. 2006. Motif search in graphs: application to metabolic networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 3, 360–368.

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

Narayanan, M., and Karp, R.M. 2007. Comparing protein interaction networks via a graph match-and-split algorithm. *J. Comput. Biol.* 14, 892–907.

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

Ogata, H., Fujibuchi, W., Goto, S., and Kanehisa, M. 2000. A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. *Nucleic Acids Res.* 28, 4021–4028.

Ostrand, P. 1970. Systems of distinct representatives II. *J. Math. Anal. Appl.* 32, 1–4.

Peri, S., Navarro, J.D., Amanchy, R., et al. 2003. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res.* 13, 2363–2371.

Pinter, R.Y., Rokhlenko, O., Yeger-Lotem, E., et al. 2005. Alignment of metabolic pathways. *Bioinformatics* 21, 3401–3408.

Reguly, T., Breitkreutz, A., Boucher, L., et al. 2006. Comprehensive curation and analysis of global interaction networks in *Saccharomyces cerevisiae. J. Biol.* 5, 11.

Rual, J.F., Venkatesan, K., Hao, T., et al. 2005. Towards a proteome-scale map of the human protein-protein interaction network. *Nature* 437, 1173–1178.

Ruepp, A., Brauner, B., Dunger-Kaltenbach, I., et al. 2008. Corum: the comprehensive resource of mammalian protein complexes. *Nucleic Acids Res.* 36, D646–D650.

Scott, J., Ideker, T., Karp, R. M., et al. 2006. Efficient algorithms for detecting signaling pathways in protein interaction networks. *J. Comput. Biol.* 13, 133–144.

SGD Project. 2008. Saccharomyces genome database. Available at: www.yeastgenome.org/. Accessed December 1, 2009.

Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. 2003. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Gen. Res*. 13, 2498–2504.

Sharan, R., Dost, B., Shlomi, T., et al. 2008. QNet: a tool for querying protein interaction networks. *J. Comput. Biol.* 15, 913–925.

Sharan, R., Ideker, T., Kelley, B.P., et al. 2005. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *J. Comput. Biol.* 12, 835–846.

Shlomi, T., Segal, D., Ruppin, E., et al. 2006. QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinform.* 7, 199.

Sohler F., and Zimmer, R. 2005. Identifying active transcription factors and kinases from expression data using pathway queries. *Bioinformatics* 21, ii115–ii122.

Stanyon, C.A., Liu, G., Mangiola, B.A., et al. 2004. A Drosophila protein-interaction map centered on cell-cycle regulators. *Genome Biol.* 5, R96.

Stelzl, U., Worm, U., Lalowski, M., et al. 2005. A human protein-protein interaction network: a resource for annotating the proteome. *Cell* 122, 957–968.

Tohsato, Y., Matsuda, H., and Hashimoto, A. 2000. A multiple alignment algorithm for metabolic pathway analysis using enzyme hierarchy. *Proc. 8th Int. Conf. Intelligent Syst. Mol. Biol. (ISMB 2000)*, 376–383.

Ulitsky I., and Shamir, R. 2007. Identification of functional modules using network topology and high-throughput data. *BMC Syst. Biol.* 1, 8.

Xenarios, I., Salwnski, L., Higney, J.X.P., et al. 2002. DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.* 30, 303–305.

Yang Q., and Sze, S.-H. 2007. Path matching and graph matching in biological networks. *J. Comput. Biol.* 14, 56–67.

Yosef, N., Kupiec, M., Ruppin, E., et al. 2009. A complex-centric view of protein network evolution. *Nucleic Acids Res.* 37, e88.

Yu, H., Braun, P., Yildirim, M., et al. 2008. High-quality binary protein interaction map of the yeast interactome network. *Science* 322, 104–110.

Zheng, Y., Szustakowski, J.D., Fortnow, L., et al. 2002. Computational identification of operons in microbial genomes. *Genome Res.* 12, 1221–1230.

Address correspondence to:
Sharon Bruckner School of Computer Science
Tel Aviv University
69978 Tel Aviv
Israel

E-mail: bruckner@tau.ac.il