Querying Graphs in Protein-Protein Interactions Networks using Feedback Vertex Set

Guillaume Blin, Florian Sikora, Stéphane Vialette

Querying Graphs in Protein-Protein Interactions Networks using Feedback Vertex Set

Guillaume Blin, Florian Sikora, Stéphane Vialette

Abstract

Recent techniques increase rapidly the amount of our knowledge on interactions between proteins. The interpretation of these new information depends on our ability to retrieve known sub-structures in the data, the Protein-Protein Interactions (PPI) networks. In an algorithmic point of view, it is an hard task since it often leads to NP-hard problems. To overcome this difficulty, many authors have provided tools for querying patterns with a restricted topology, *i.e.* paths or trees in PPI networks. Such restriction leads to the development of fixed parameter tractable (FPT) algorithms, which can be practicable for restricted sizes of queries. Unfortunately, GRAPH HOMOMORPHISM is a W[1]-hard problem, and hence, no FPT algorithm can be found when patterns are in the shape of general graphs. However, Dost *et al.* [2] gave an algorithm (which is not implemented) to query graphs with a bounded treewidth in PPI networks (the treewidth of the query being involved in the time complexity). In this paper, we propose another algorithm for querying pattern in the shape of graphs, also based on dynamic programming and the color-coding technique. To transform graphs queries into trees without loss of informations, we use feedback vertex set coupled to a node duplication mecanism. Hence, our algorithm is FPT for querying graphs with a bounded size of their feedback vertex set. It gives an alternative to the treewidth parameter, which can be better or worst for a given query. We provide a python implementation which allows us to validate our implementation on real data. Especially, we retrieve some human queries in the shape of graphs into the fly PPI network.

Index Terms

Graph Query, Pattern-Matching, Dynamic Programming, Protein-Protein Interactions networks.

I. INTRODUCTION

CONTRARY to what was predicted years ago, the human genome project has highlighted that human complexity may not only rely on its genes (only 25 000 for human compared to the 30 000 and 45 000 for the mouse and the poplar respectively). This observation increased the interest in protein properties (*e.g.* their numbers, functions, complexity and interactions). Among other protein properties, the set of all their interactions for an organism, called Protein-Protein Interactions (PPI) networks, have recently attracted lot of interest. The number of reported interactions increases rapidly due to the use of various genome-scale screening techniques [3], [4], [5]. Unfortunately, acquiring such valuable resources is prone to high noise rate [3], [6].

Comparative analysis of PPI tries to determine the extent to which protein networks are conserved among species. Indeed, it was observed that proteins functioning together in a pathway (*i.e.*, a path in the interactions graph) or a structural complex (*i.e.*, an assembling of strongly connected proteins) are likely to evolve in a correlated fashion and during evolution, all such functionally linked proteins tend to be either preserved or eliminated in a new species [7].

In this article, we focus on the following related problem called GRAPH QUERY (formally defined later). Given a PPI network and a pattern with a graph topology, find a subnetwork of the PPI network that is as similar as possible to the pattern, in respect to the initial topology. Similarity is measured both in terms of sequence similarity and graph topology conservation.

Université Paris-Est, LIGM - UMR CNRS 8049, France.
E-mail: {gblin, sikora, vialette}@univ-mlv.fr

An extended abstract of this work appeared in Proceedings of the 5^{th} International Symposium on Bioinformatics Research and Applications (ISBRA'09) [1].

Unfortunately, this problem is clearly equivalent to the NP-complete subgraph homeomorphism problem [8]. Recently, several techniques have been proposed to overcome the difficulty of this problem. By restricting the query to a path of length less than five, Kelley *et al*. [9] developped PathBlast, a software with a factorial time complexity which allows one consecutive mismatch. Later on, Shlomi *et al*. [10] proposed an alternative, called QPath, for querying paths in a PPI network which is based on the color-coding technique introduced by Alon, Yuster and Zwick [11]. The use of this technique allows to define a fixed-parameter tractable (FPT) algorithm parameterized by the size of the query. Recall that a parameterized problem is FPT if it can be determined in $f(k)n^{\mathcal{O}(1)}$ time, where f is a function only depending on the parameter k, and n is the size of the input [12]. In addiction of being faster, QPath deals with longer paths (until size ten) and allows more flexibility by considering a bounded number of non-exact matches.

By restricting the query to a tree, Pinter *et al*. [13] proposed an algorithm that is restricted to forest PPI networks (*i.e.*, collection of trees). Finally, Dost *et al*. [2] developed QNet, an algorithm to handle tree query in the general context of PPI networks. The authors also gave some theoretical results for querying graphs using the tree decomposition of the query.

Since QNet is the main reference in this field and is quite related to the work presented in this paper, let us present it briefly. QNet is an FPT algorithm for querying trees in a PPI network. The time complexity is $2^{\mathcal{O}(k)}m\ln(\frac{1}{\epsilon})$, where k is the number of proteins in the query, m the number of edges of the PPI network and $1-\epsilon$ the success probability (for any $\epsilon > 0$). As QPath, QNet uses dynamic programming together with the color-coding technique. For querying graphs in a network, QNet uses, as a subroutine, an algorithm to query trees. To do so, they perform a tree decomposition (a formal definition of a tree decomposition can be found in [14]). Roughly speaking, it is a transformation of a graph into a tree, a tree node (or a bag) can contain several graph nodes. There exists several algorithms to perform such a transformation. The *treewidth* of a graph is the minimum (among all decompositions) of the cardinality of the largest bag minus one. Computing the treewidth is, however, NP-Hard [15]. From this tree decomposition, the time complexity of QNet is $2^{\mathcal{O}(k)}n^{t+1}\ln(\frac{1}{\epsilon}))$ time, where k is the size of the query, n is the size of the PPI network, t is the treewidth of the query, and $1-\epsilon$ is the success probability (for any $\epsilon > 0$).

QNet is an algorithm for querying trees in a PPI network. A logical extension would be to query graphs. The authors of [2] provide a theoretical solution, without implementation and which depends on the treewidth of the query. We propose here an alternative solution, that uses the color-coding technique (Section II). We provide in Section III some experimental results.

II. PADA1 as an alternative to QNet

In this section, we propose an alternative to QNet called PADA1 (Protein Alignment Dealing with grAphs). At a more general level, QNet and PADA1 use the very same approach: transform the query into a tree and find an occurrence of that tree in the PPI network by dynamic programming. However, whereas QNet uses tree decompositions, PADA1 combines feedback vertex sets together with nodes duplications (Algorithm Graph2Tree). Let note that independently, Cheng *et al.* use a similar technique to transform a graph into a tree in [16] in order to query a graph in a network. However, unlike our approach, in [16] the authors do not use nodes duplication and hence, it is not clear they can ensure that all the edges of the query are kept in the results (especially the ones linking the nodes belonging to the feedback vertex set). It is worth mentioning that, following the example of QPath and QNet, we will consider non-exact matches (*i.e.*, allowing indels). Since we allow queries to be graphs, PADA1 is clearly an extension of QPath and a real alternative to QNet.

A. Transforming the query into a tree

Let us first present Algorithm Graph2Tree which transforms a graph $G = (V, E)$ into a tree. Our transformation is lossless, and hence, one can reconstruct the graph starting from the tree. The main idea of Algorithm Graph2Tree is to transform the graph into a tree by iteratively finding a cycle C, duplicating a node of C, and finally breaking cycle C by deleting one of its edges (see Figure 1 for an

illustration on how to break a cycle at vertex v_1). Central in our approach is thus the node duplication procedure (Algorithm DUPLICATE). For each $u \in V$, write $d(u)$ for the set of all copies of vertex u including itself and $N(u)$ for the set of all its neighbors.

```
1  Function GRAPH2TREE(G)
2  begin
3      for all u of V, d(u) ← {u};
4      FVS ← FEEDBACKVERTEXSET(G);
5      while ∃ a cycle C in G do
6          Let u be one vertex randomly chosen in C ∩ FVS;
7          Let v be one vertex randomly chosen in C ∩ N(u);
8          DUPLICATE(G, v, u, d);
9      end
10 end
```

Algorithm 1: GRAPH2TREE algorithm

Let F denote the set of all nodes of G that have been duplicated at the end of Algorithm GRAPH2TREE, *i.e.*, $F = \{v \in V : |d(v)| > 1\}$. The cardinality of F turns out to be an important parameter since, as we will prove soon, the overall time complexity of PADA1 mostly depends on $|F|$ and not on the total number of duplications. Minimizing the cardinality of F is the well-known NP-complete FEEDBACK VERTEX SET problem [17]: Given a graph G, find a minimum cardinality subset of vertices with the property that removing of these vertices from the graph results in an acyclic graph.

We have implemented a "brute-force" algorithm for the FEEDBACK VERTEX SET problem. Once this set is computed, we duplicate a node of a cycle as long as there is a cycle in the graph. By definition of the feedback vertex set, there is at least one node in a cycle belonging to the feedback vertex set. The time complexity of the Algorithm GRAPH2TREE is dominated by the process of computing the feedback vertex set, which is in $\mathcal{O}(2^{|V|} \times |E|)$ since there are $2^{|V|}$ potential subgraphs. Nevertheless this solution is practical since it is still running in seconds if $|V|$ is smaller than twenty. Indeed, the overall complexity of PADA1 considerably limits the size of our graph query. However, one may also consider an efficient FPT algorithm such as the one of Guo *et al.* [18], using *iterative compression*, in addition of the quadratic kernalization of Thomassé [19] in order to compute efficiently the feedback vertex set.

B. Tree matching

We now assume that the query has been transformed into a tree (with duplicated nodes) by Algorithm GRAPH2TREE, and hence we only consider tree queries from this point. We show that an occurrence of such a tree can be found in a PPI network by dynamic programming.

```
1  Function FEEDBACKVERTEXSET(G = (V, E))
2  begin
3      for (i = 0 ; i < |V| ; i++) do
4          foreach subgraph G' = (V', E') of G such that |V'| = |V| − i do
5              if G' is acyclic then
6                  return V \ V';
7              end
8          end
9      end
10 end
```

Algorithm 2: Compute the Feedback Vertex Set of a graph.

```
Function DUPLICATE(G = (V, E), v_a, v_b, d)
begin
    Let i ← |d(v_b)|;
    Let v_{b,i} be a new node;
    V ← V ∪ {v_{b,i}};
    d(v_b) ← d(v_b) ∪ {v_{b,i}} ;
    E ← E − {(v_a, v_b)};
    E ← E ∪ {(v_a, v_{b,i})};
end
```

Algorithm 3: Algorithm to duplicate a node when a cycle is detected.

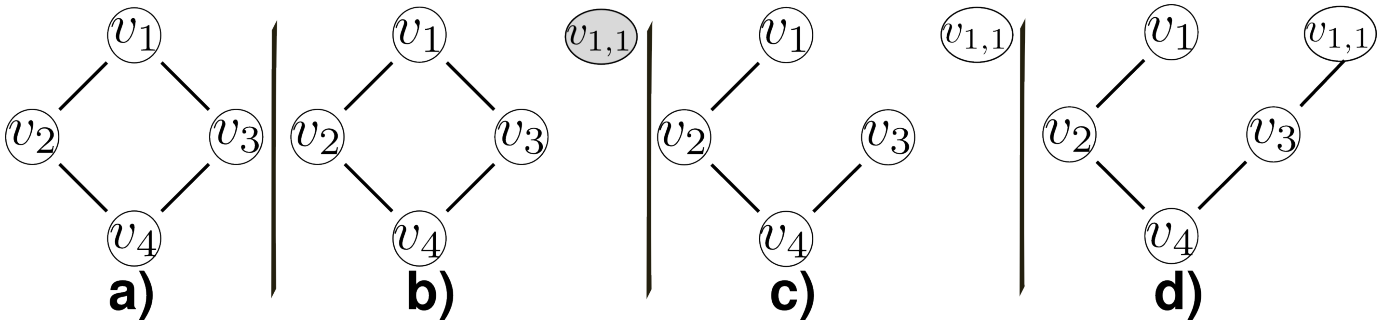

Figure 1. Steps when Duplicate(G, v_3, v_1, d) is called on graph a). b) A node $v_{1,1}$ from v_1 is created. c) The edge (v_3,v_1) is deleted: the cycle is then broken. d) The edge (v_3,$v_{1,1}$) is added. Finally, the resulting graph is acyclic, and $d(v_1) = \{v_1, v_{1,1}\}$.

Let us fix notations. PPI networks are represented by undirected edge weighted graphs $G_N = (V_N, E_N, w)$; each node of V_N represents a protein and each weighted edge $(v_i, v_j) \in E_N$ represents an interaction between two proteins. A query is given by a tree $T_Q = (V_Q, E_Q)$ (output of Algorithm GRAPH2TREE on the graph query). The set V_Q represents proteins while E_Q represents interactions between these proteins. As in QNet, we do not give weight on the query. Indeed, in PPI networks, the weight represents probabilities of the interactions. There is no clue to use those probabilities in the query. In the following, we will consider that T_Q is an ordered tree where $q_1, \ldots, q_{n_q}$ denote the ordered n_q children of any node q. As we will show afterwards, the result does not depend on this ordering.

Let $h(p_1, p_2)$ be a function that returns a similarity score between two proteins p_1 and p_2. The similarity considered here will be computed according to amino-acid sequences similarity (using BLASTp [20]). In the following, given two nodes v_1 and v_2 of V_Q (or V_N), we write $h(v_1, v_2)$ for the similarity between the two proteins corresponding to v_1 and v_2. A node v_1 is considered to be *homologous* to a node v_2 if the corresponding similarity score $h(v_1, v_2)$ is above a given threshold. Biologically, one can assume that two homologous proteins have probably similar functions. Clearly, for every node v of F, all nodes in $d(v)$ are homologous with the same protein.

An *alignment* of the query T_Q and G_N is defined as: (i) a subgraph $G_A = (V_A, E_A, w) \subseteq G_N = (V_N, E_N, w)$, such that $V_A \subseteq V_N$ and $E_A \subseteq E_N$, and (ii) a mapping $\sigma : V_Q \to V_A \cup \{\texttt{del}\}$. More precisely, the function σ is defined such that if $\sigma(q) = v$ then q and v are homologous.

For a given alignment of T_Q and G_N, a node q of V_Q is said to be *deleted* if $\sigma(q) = \texttt{del}$ and *matched* otherwise. Moreover, any node v_a of V_A such that $\sigma^{-1}(v_a)$ is undefined is said to be *inserted*. Note that, similarly to QNet, only nodes of degree two can be deleted (we can only contract paths). For practical applications, the number of insertions (resp. deletions) is limited to be at most N_{ins} (resp. N_{del}), each involving a penalty score δ_i (resp. δ_d).

The GRAPH QUERY problem can be thus defined as follow: Given a query T_Q with duplicated nodes, a PPI network G_N, a similarity function h, penalty scores δ_i and δ_d for insertions and deletions, find an

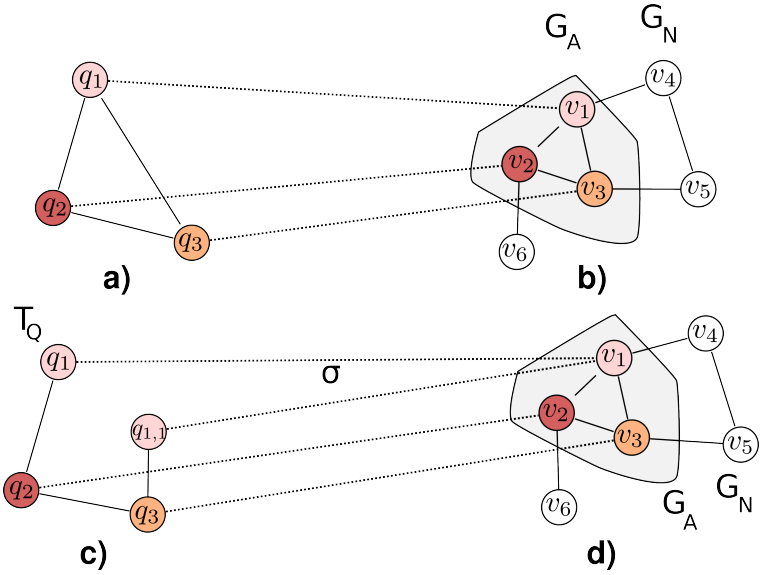

Figure 2. a) The graph query with a cycle, before calling GRAPH2TREE algorithm. c) The query after calling GRAPH2TREE where q_1 has been duplicated. Thus, q_1 and $q_{1,1}$ have to be aligned with the same node of the network. b) and d) denote the resulting graph alignment G_A, subgraph of the network G_N. The horizontal dashed lines denote a match between two proteins.

alignment (G_A, σ) between T_Q and G_N of maximum score. The score of an alignment is defined as the sum of (i) similarity scores of aligned nodes (*i.e.*, $\sum_{\substack{v \in V_A \\ \sigma^{-1}(v) \text{ defined}}} h(v, \sigma^{-1}(v))$), (ii) the sum of weights of edges in E_A (*i.e.*, $\sum_{e \in E_A} w(e)$), (iii) a penalty score δ_d for each node deletion (*i.e.*, $\sum_{\substack{q \in V_Q \\ \sigma(q) = \text{del}}} \delta_d$), and (iv) a penalty score δ_i for each node insertion (*i.e.*, $\sum_{\substack{v \in V_A \\ \sigma(v)^{-1} \text{ undefined}}} \delta_i$).

The general problem is NP-complete. However, it is Fixed Parameter Tractable in case the query is a tree by a combination of the *color-coding* technique [11] and dynamic programming. This randomized technique allows to find simple paths of length k in a network in $\mathcal{O}(2^k)$ time (to be compared to the $\mathcal{O}(n^k)$ time brute-force algorithm), where n is the number of proteins in the network [21]. In [2], the authors of QNet adapted this technique for their query algorithm. Since one is looking for an alignment, each node of the query has to be considered once (and only once) in an incremental build of the alignment by dynamic programming. Thus, one has to maintain a list of the nodes already considered in the query. Therefore, on the whole, one has to consider all $\mathcal{O}(n^k)$ potential alignments, with $n = |V_N|$ and $k = |V_Q|$.

Using color-coding, one may decrease this complexity to $\mathcal{O}(2^k)$. First, nodes of the network are colored randomly using k colors, where $k = |V_Q|$. Then, looking for a colorful alignment (*i.e.*, an alignment that contains each color once) leads to a solution, which is not necessarily optimal. Therefore, one only needs to maintain a list of the colors already used in the alignment, storable in a table of size in $\mathcal{O}(2^k)$. In order to get an optimal solution, this process is repeated. More precisely, according to QNet [2], since a colorful alignment happens with probability $\frac{k!}{k^k} \simeq e^{-k}$, the coloration step has to be done $\ln(\frac{1}{\epsilon})e^k$ times to obtain an optimal alignment with high probability ($1 - \epsilon$, for any ϵ).

The QNet dynamic programming algorithm can be summarized as follows. By an incremental construction, for each $(q_i, q_j) \in E_Q$ when one is considering $q_i \in V_Q$ aligned with a node $v_i \in V_N$, check whether the score of the alignment is improved through: (i) a match of q_j and any v_j of V_N such that q_j and v_j are homologous and $(v_i, v_j) \in E_N$, (ii) an insertion of a node v_j of V_N in the alignment graph G_A, and (iii) a deletion of q_j. This is done for a given coloration of the network, and repeated for each coloration.

Hereafter, we define an algorithm, inspired from QNet, which consider a query tree T_Q, a PPI network G_N and seeks for an alignment (G_A, σ). It is worth noticing that for a given coloration, our algorithm, as QNet, is exact. To deal with duplicated nodes (cf. GRAPH2TREE algorithm), we pre-compute all possible assignment, called A, of the duplicated nodes V_Q of T_Q. More precisely, $\forall q \in F$, $\forall v \in V_N$, define

```
Function PADA1 (T_Q, G_N, h, threshold)
begin
    BestG_A ← ∅; BestScore ← −∞;
    for (i = 0; i < ln(1/ε)e^k; i++) do
        randomly colorize G_N with k + N_ins colors;
        foreach valid assignment A do
            ScoreG_A ← BestConstraintAlignment(G_N, T_Q, A, h, threshold) + score(A);
            if ScoreG_A > BestScore then
                Save coloration, A;
                BestScore ← ScoreG_A;
            end
        end
    end
    Load coloration, A;
    return Backtracking();
end
```

Algorithm 4: Sketch of the PADA1 algorithm to align a query graph to a network.

$\sigma(q') = v \ \forall q' \in d(q)$. We then compute for each assignment A the score of an alignment with respect to A. We denote BESTCONSTRAINTALIGNMENT this step. The difficulty lies in the construction of the best alignment by dynamic programming, with respect to A.

As done in QNet, we use a set S_C of $k + N_{ins}$ colors (as needed by color-coding) which will be used when a node is matched or inserted. Moreover, in order to deal with potential duplicated nodes in T_Q, we have to use another multiset S of colors (*i.e.*, the colors in this set can appear more than once), rather than a classical set as in QNet. Indeed, every node in $d(q)$ such that $q \in F$, must use the same color.

As a preprocess to PADA1, G_N can be pruned, as shown in [22]. Let $u \in G_N$ be a protein which is not homologous with any protein of the query and $v \in G_N$ be a protein which is homologous with a protein of the query. Then, u can be too far from any v in terms of shortest path length to be inserted in the solution in regards to the maximum number of insertions (*i.e.*, N_{ins}). According to this remark, u is kept in G_N only if there are two proteins v_1 and v_2, both homologous with a protein of the query, such that $dist(u, v_1) + dist(u, v_2) \leq N_{ins} + 1$, where $dist(u, v)$ is the length of the shortest path between u and v. Otherwise, u can never been in a solution, and hence can be safely deleted from G_N.

Once G_N has been pruned, PADA1 can be launched for each *valid* connected component of G_N. A component is said to be valid if it contains at least $k - N_{del}$ proteins which are homologous with a protein of the query, where k is the size of the query. Otherwise, a solution can never be found in this component, and hence there is no need to consider it. As stated in [22], there is in practice only 5% of the network proteins which are on average homologous with a query protein.

Algorithm 4 may be summarized as follow. Perform $\ln(\frac{1}{\epsilon})e^k$ random colorations of the PPI network G_N to ensure optimality with a probability of at least $1 - \epsilon$. The coloration consists in defining a function $c : V_N \rightarrow S_C$, where the color in S_C is randomly chosen. Then, for each coloration, we build all possible valid assignments A of the duplicated nodes. An assignment A is valid if no two non homologous nodes are matched in A. For each such assignment A, we compute the best score of an alignment according to A with Algorithm BESTCONSTRAINTALIGNMENT. We keep the best score of these trials and obtain the corresponding alignment by a classic backtracking technique. The score of the assignment of the duplicated nodes is computed separatly as follows. Indeed, in order not to take into account the homology score of q more than once (*i.e.* $|d(q)|$ times) in the overall score (namely $ScoreG_A$), we precompute the part of the score induced by the duplicated nodes (namely $score(A)$).

$$score(A) = \sum_{\substack{q \in F \\ A(q) \neq \texttt{del}}} h(q, A(q)) + \sum_{\substack{q \in F \\ A(q) = \texttt{del}}} \delta_d$$

Let us now describe in details the BESTCONSTRAINTALIGNMENT step that returns the score of G_A according to the precomputed assignment A.

$$ScoreG_A \leftarrow \max_{v \in V_N} W^M(root, v, S, 1, A) + score(A)$$

The best alignment score is obtained by finding among all possibilities the best way to align the *root* of the query to any protein v of the network. Similarly to QNet, the root is selected arbitrarily, but it is always a node of degree one. Moreover, the score is computed only if the *root* and v are homologous. In this initial step, S represents the multiset of colors defined previously. As in QNet, for each query node q, let denote by $q_1, q_2, \ldots, q_{n_q}$ its n_q children. To obtain the best alignment, we use three tables, namely W^M, W^I and W^D, which are filled as follows.
If $|S| \leq 1$,

$$W^M(q, v, S, j, A) \leftarrow -\infty$$

Else,

$$W^M(q, v, S, j, A) \leftarrow \max_{\substack{u:(u,v) \in E_N \\ S' \subset S \\ c(v) \in S' \\ c(u) \in S - S'}} W^M(q, v, S', j-1, A) + \begin{cases} \text{(* Matching, child } j \text{ *)} \\ W^M(q_j, u, S - S', n_{q_j}, A) + w(u, v), \\ \\ \text{(* Insertion, node } u \text{ *)} \\ W^I(q_j, u, S - S', A) + w(u, v), \\ \\ \text{(* Deletion, child } j \text{ *)} \\ W^D(q_j, v, S - S', A) \end{cases}$$

In the computation of $W^M(q, v, S, j, A)$, we consider that q is already aligned with v. Thus, the value stored in $W^M(q, v, S, j, A)$ is the maximum score of the subtree rooted at q and considering only its j first children. This score corresponds to the sum of the score of the subtree rooted at q considering only its $j-1$ first children (*i.e.*, $W^M(q, v, S', j-1, A)$) and the score for the best aligment of the j^{th} child of q – denoted q_j. Indeed, when considering q_j one can either (1) match it with a neighbor u of v, (2) delete it or (3) insert a neighbor u of v in the alignment. In the dynamic programming equation, we denote by n_{q_j} the number of children of q_j.

To obtain the optimal solution, each subset of the multiset S of colors has to be considered. In others words, we consider each subset of colors used for the first $j-1$ subtrees of q (the subset S'), and therefore each subset of colors used for the j^{th} subtree (the subset $S - S'$). In case S is a singleton, the score is $-\infty$ since there are at least $j+1$ nodes (*i.e.,* q and the nodes of the j subtrees rooted at $q_1, q_2, \ldots q_j$) to add in the solution with only one color, while the deletion of nodes with degree one is forbidden.

$$W^M(q, v, S, 0, A) \leftarrow \begin{cases} h(q, v) \text{ if } |d(q)| = 1, \\ -\infty \text{ if } A(q) \neq v, \\ 0 \text{ else (* } |d(q)| > 1 \text{ and } A(q) = v \text{ *)} \end{cases}$$

Entries corresponding to $W^M(q, v, S, 0, A)$ are the base cases of the recursion. When q is not a duplicated node (*i.e.*, $|d(q)| = 1$), the value is simply the similarity score with v given by h. Otherwise, the assignment of q has already been defined in A and taken into account in $score(A)$, and has to be preserved; otherwise, we return $-\infty$ to forbid such an alignment.

$$W^I(q,v,S,A) \leftarrow \begin{cases} \text{if } A^{-1}(v) = \emptyset \text{ and } |S| > 1 \\ \quad \max_{\substack{u:(u,v)\in E_N \\ c(u)\in S-\{c(v)\}}} \begin{cases} W^M(q,u,S-\{c(v)\},n_q,A) + w(u,v) + \delta_i, \\ W^I(q,u,S-\{c(v)\},A) + w(u,v) + \delta_i, \end{cases} \\ \text{else} \\ \quad -\infty \end{cases}$$

When computing $W^I(q,v,S,A)$, the node v is considered to be inserted. Thus, the computation will continue with a neighbor u of v, performing either a match or another insertion.

In order to be coherent with A, if v has to be aligned with a duplicate node (*i.e.*, $A^{-1}(v) \neq \emptyset$), inserting v is forbidden.

$$W^D(q,v,S,A) \leftarrow \begin{cases} \text{if degree(q)} \neq 2 \\ \quad -\infty \\ \text{else if } |d(q)| = 1 \\ \quad \max_{u:(u,v)\in E_N} \begin{cases} W^M(q_1,u,S,n_{q_1},A) + w(u,v) + \delta_d, \\ W^I(q_1,u,S,A) + w(u,v) + \delta_d, \\ W^D(q_1,v,S,A) + \delta_d \end{cases} \\ \text{else} \\ \quad \text{if } A(q) = \texttt{del} \\ \quad\quad \max_{u:(u,v)\in E_N} \begin{cases} W^M(q_1,u,S,n_{q_1},A) + w(u,v), \\ W^I(q_1,u,S,A) + w(u,v), \\ W^D(q_1,v,S,A) \end{cases} \\ \quad \text{else} \\ \quad\quad -\infty \end{cases}$$

Finally, when computing $W^D(q,v,S,A)$, the node v and the father of q are considered as aligned and the node q to be deleted. The alignment then continues from q_1, the only son of q since deletion is only allowed for nodes with degree two. If q is a duplicated node (*i.e.*, $|d(q)| > 1$), the deletion is only allowed when $A(q) = \texttt{del}$ to respect the reference assignment A. Again, in this case, the cost of a deletion δ_d is already counted in $score(A)$.

Let us note that since in the dynamic programming, for each j s.t. $1 \leq j \leq n_q$, all set $S' \subset S$ are considered, hence the result does not depend on the ordering of T_Q.

Let us now analyze the complexity of PADA1. The whole complexity depends essentially on lines 5 to 12. Let us consider the complexity of one iteration (we have $\ln(\frac{1}{\epsilon})e^k$ iterations). The random coloration can be done in $\mathcal{O}(n)$, where $n = |V_N|$. There are $n^{|F|}$ possible assignments in the worst case (*i.e.*, if all the proteins in F are homologous with the n proteins of the network). The complexity of BESTCONSTRAINTALIGNMENT is $2^{\mathcal{O}(k+N_{ins})}mN_{del}$ as in QNet, where k is the size of the graph query and $m = |E_N|$, since our modifications are essentially additional tests which can be done in constant time.

Let us note that the complexity of GRAPH2TREE is negligible compared to the overall complexity of Algorithm PADA1. Indeed, the complexity of Algorithm GRAPH2TREE only depends on the query size k, with $k \ll n$. Therefore, on the whole, the complexity of PADA1 is $\mathcal{O}(n^{|F|}2^{\mathcal{O}(k+N_{ins})}mN_{del}\ln(\frac{1}{\epsilon}))$ time for any desired success probability $1-\epsilon$ (with $\epsilon > 0$). Observe that the time complexity does not depends on the total number of duplicated nodes (*i.e.*, $\sum_{q\in F}|d(q)|$), but on the size of F.

III. EXPERIMENTAL RESULTS

According to the authors of QNet, one may query a PPI network by running an $2^{\mathcal{O}(k)}n^{t+1}$ time algorithm $\ln(\frac{1}{\epsilon})e^k$ times, where t is the treewidth of the query graph. Thus, the difference between the two algorithms is mainly related to the "$t+1$ versus $|F|$" question (where $|F|$ is the size of the set of families of

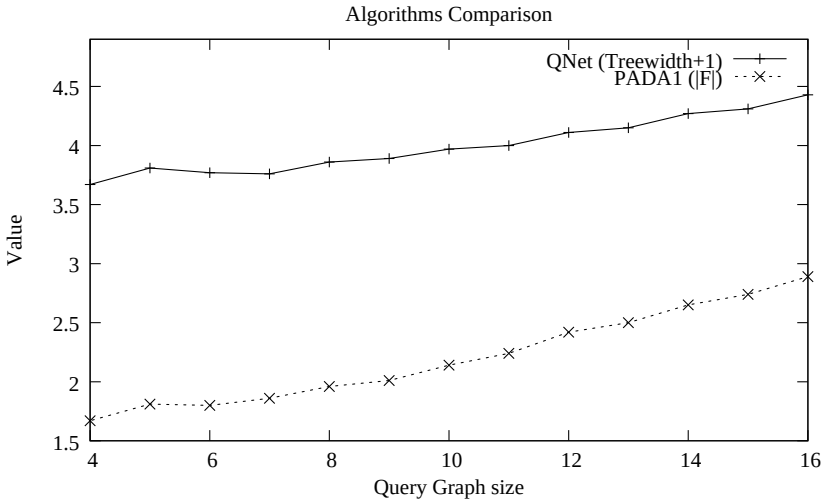

Figure 3. Comparison between QNet (*i.e.*, the treewidth+1 value) and PADA1 parameter (*i.e.*, the size of the feedback vertex set). In QNet (resp. in PADA1), the treewidth (resp. the feedback vertex set) is computed over the query. Here the size of the query graph correspond to the number of nodes (there are usually between five and fifteen proteins in a classical query).

duplicated nodes computed by Algorithm GRAPH2TREE). Let us recall that the size of $|F|$ given by our algorithm is equal to the feedback vertex set size of the query graph. According to Bodlaender and Koster [23], the treewidth of a graph is at most equal to the feedback vertex set plus one. However, it only gives an upper-bound. We have conducted some experimental tests to compare these two parameters in practice on random graphs. The method is as follows: for each different size of graph (the number of nodes varies from 4 to 16 while the number of edges varies from $|V|$ to $2.|V|$), we get the average treewidth and feedback vertex set values over 30000 connected graphs, randomly constructed with the NetworkX library (http://networkx.lanl.gov/). Treewidth is computed with the exact algorithm provided by http://www.treewidth.com/, while the size of the feedback vertex set is computed with our GRAPH2TREE algorithm. Results in Figure 3 suggest that parameter $|F|$ is usually smaller for moderate size graphs (*i.e.*, those query graphs for which PADA1 and QNET are still practicable). In summary, PADA1 is an alternative to the QNet algorithm, with a different parameter involved in the time complexity. One can determine which parameter is more suitable for a given query and then uses the appropriate algorithm.

In practice, our upper-bound is largely over estimated. Indeed, each element of F must be assigned to a different node of the network, and hence, there are less than n possibilities for each element of F. The worst number of runs of BESTCONSTRAINTALIGNMENT is $\frac{n!}{(n-|F|)!}$, the number of combinations.

Moreover, we only consider valid assignments and there are only few such assignments. Indeed, a protein is, on average, homologous to dozens of proteins, which is quite less than the number of proteins in a classical PPI network (*e.g.* $n \simeq 5.000$ for the yeast). For example, if $|F| = 3$ and if the protein represented by this unique element of F is homologous to ten proteins in the PPI network, then, the number of assignment will not be n^3 but only 10^3. Here, the running time is largely reduced. Therefore, and not surprisingly, the BLAST threshold used to determine if a protein is homologous to another has a huge impact on the running time of the algorithm.

Finally, observe that in QNet, for a given treewidth, the query graph can be very different. For example, in the resulting tree decomposition of the graph, there is no limit on the number of bags of size t. Furthermore, in a given bag, the topology is arbitrary (*e.g.*, a clique), potentially requiring an exhaustive enumeration upper-bounded by n^{t+1}. Therefore, the treewidth value does not indicate how many times an exhaustive enumeration has to be done.

We would have liked to compare in practice our algorithm to QNet, but, unfortunately, their version querying graphs is not yet implemented. Comparing our algorithm for simple trees queries with QNet would not make sense since PADA1 is not optimized for this special case.

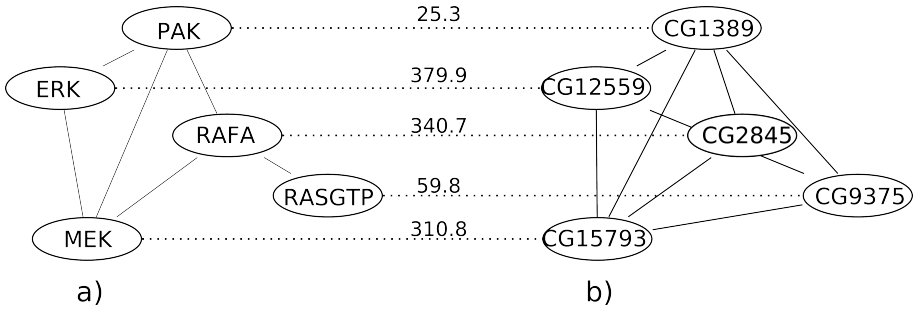

Figure 4. A result sample of our algorithm. a) A MAPK human query, get from [26], with three cycles. b) The alignment graph given by our algorithm in the fly PPI network. Dashed lines denotes the BLAST homology scores between the two proteins. Our algorithm retrieves a query graph in an other network. As in QNet [2], it seems to be that there is some conservation between these two species.

In order to validate our algorithm, we perform the experimental tests on real data proposed by QNet [2]. In our experiments, the data for the PPI network of the fly and the yeast have been obtained from the DIP database[1][24]. The yeast network contains 4 738 proteins and 15 147 interactions, whereas the fly network contains 7 481 proteins and 26 201 interactions.

The first experiment consists in retrieving trees. To do so, the authors of QNet extract randomly trees queries of size 5 to 9 from the yeast network and try to retrieve them in this network. Each query is modified with at most two insertions or deletions. We also have successfully retrieved these queries.

The second experiment was performed across species. The Mitogen-Activated Protein Kinase (MAPK) are a collection of signal transduction queries. According to [25], they have a critical function in the cellular response to extracellular stimuli. They are known to be conserved through different species. We obtained the human MAPK from the KEGG database [26] and tried to retrieve them in the fly network as done in QNet. While QNet uses only trees, we were able to query graphs. The results were satisfying since we retrieved them, with few or without modifications. The Figure 4 shows a sample of our results on real data. This suggests a potential conservation of patterns across species.

The BLAST threshold have deep impact on the running time. Moreover, we could certainly speed-up the running time by stopping earlier the dynamic programming. Indeed, one can stop if there are more nodes to color than the number of available colors in a step of the dynamic programming. This trick implies to look for the available number of deletions and insertions. Thus, for the moment we only stop when there is only a single color available. Another improvement can be to switch the coloration step with choices of assignements A. A final speed-up possibility can be to use the Hüffner *et al.* technique [27], which basically consists in increasing the number of colors used during the coloration step.

IV. Conclusion

In this paper, we have tried to improve our understanding in PPI networks by developing a tool called PADA1 (available uppon request), to query graphs in PPI networks. This algorithm, which is FPT for query graphs with a constant size of their feedback vertex set, has a time complexity of $\mathcal{O}(n^{|F|}2^{\mathcal{O}(k+N_{ins})}mN_{del}\ln(\frac{1}{\epsilon}))$, where n (resp. m) is the number of nodes (resp. edges) in the PPI network, k is the number of nodes in the query, N_{ins} (resp. N_{del}) is the maximum number of insertions (resp. deletions) allowed, ϵ is any value > 0 such that $1-\epsilon$ is the desired success probability, and $|F|$ is the minimum number of nodes which have to be duplicated to transform the query graph into a tree (solving the Feedback Vertex Set problem). This last parameter is the main difference with QNet of Dost *et al.* [2], which uses the treewidth of the query (unimplemented algorithm). Consequently, PADA1 is an alternative to QNet and one can determine which parameter is better considering a query. These algorithms both use the color coding technique and are both exact for a given coloration. We have performed some

[1]http://dip.doe-mbi.ucla.edu/

tests on real data and have retrieved known paths in the yeast PPI network. Moreover, we have retrieved known human paths in the fly PPI network.

The time complexity of our algorithm depends on the number of nodes which have to be duplicated in the graph query. This number is directly connected to the initial topology of the query graph. Obtaining more information about the topology of the queries and the average number of homologous for proteins in the query are of particular interest in this context. Future works include using these informations to predict average time complexity.

Knowing if GRAPH TOPOLOGICAL CONTAINMENT – determining if a graph G has a subgraph that is a subdivision of a parameter graph H – is FPT or W[1]-hard is still an open problem (conjectured to be FPT by Fellows [28]). In the context of proteins queries, one can ask if the query H appears in the network G with an unbounded number of insertions.

Finally, recently, a problem close to this one, called GRAPH MOTIF, has been settled by Lacroix, Fernandes and Sagot [29]. Roughly speaking, one is only concerned in find a connected occurence of the query, which is defined without a given topology, that is a set or a multiset of proteins. This new definition leads to new issues, and has already been investigated [30], [31], [32], [33], [34], [35], [22].

ACKNOWLEDGEMENT

The authors would like to thank Banu Dost for providing us the QNet source code and their test data. We also thank the anonymous reviewers for their helpful comments and suggestions for improving the manuscript.

REFERENCES

[1] G. Blin, F. Sikora, and S. Vialette, "Querying Protein-Protein Interaction Networks," in *5th International Symposium on Bioinformatics Research and Applications (ISBRA'09)*, ser. LNBI, S. Istrail, P. Pevzner, and M. Waterman, Eds., vol. 5542. Fort Lauderdale, FL, USA: Springer-Verlag, May 2009, pp. 52–62.

[2] B. Dost, T. Shlomi, N. Gupta, E. Ruppin, V. Bafna, and R. Sharan, "QNet: A Tool for Querying Protein Interaction Networks," *RECOMB*, pp. 1–15, 2007.

[3] A. Gavin, M. Boshe, *et al.*, "Functional organization of the yeast proteome by systematic analysis of protein complexes," *Nature*, vol. 414, no. 6868, pp. 141–147, 2002.

[4] Y. Ho, A. Gruhler, *et al.*, "Systematic identification of protein complexes in Saccharomyces cerevisae by mass spectrometry," *Nature*, vol. 415, no. 6868, pp. 180–183, 2002.

[5] P. Uetz, L. Giot, *et al.*, "A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisae," *Nature*, vol. 403, no. 6770, pp. 623–627, 2000.

[6] T. Reguly, A. Breitkreutz, L. Boucher, B. Breitkreutz, G. Hon, C. 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*, 2006.

[7] M. Pellegrini, E. Marcotte, M. Thompson, D. Eisenberg, and T. Yeates, "Assigning protein functions by comparative genome analysis: protein phylogenetic profiles," *PNAS*, vol. 96, no. 8, pp. 4285–4288, 1999.

[8] M. Garey and D. Johnson, *Computers and Intractability: a guide to the theory of NP-completeness*. San Franciso: W.H. Freeman, 1979.

[9] B. Kelley, R. Sharan, R. Karp, T. Sittler, D. E. Root, B. Stockwell, and T. Ideker, "Conserved pathways within bacteria and yeast as revealed by global protein network alignment," *Proceedings of the National Academy of Sciences*, vol. 100, no. 20, pp. 11 394–11 399, 2003.

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

[11] N. Alon, R. Yuster, and U. Zwick, "Color coding," *Journal of the ACM*, vol. 42, no. 4, pp. 844–856, 1995.

[12] R. Downey and M. Fellows, *Parameterized Complexity*. Springer-Verlag, 1999.

[13] R. Pinter, O. Rokhlenko, E. Yeger-Lotem, and M. Ziv-Ukelson, "Alignment of metabolic pathways," *Bioinformatics*, vol. 21, no. 16, pp. 3401–3408, 2005.

[14] H. Bodlaender, "A tourist guide through treewidth," *Acta Cybernetica*, vol. 11, pp. 1–23, 1993.

[15] S. Arnborg, D. Corneil, and A. Proskurowski, "Complexity of finding embeddings in a k-tree," *Journal on Algebraic and Discrete Methods*, vol. 8, no. 2, pp. 277–284, 1987.

[16] Q. Cheng, P. Berman, R. Harrison, and A. Zelikovsky, "Fast Alignments of Metabolic Networks," in *Proceedings of the 2008 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE Computer Society, 2008, pp. 147–152.

[17] R. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, J. Thatcher and R. Miller, Eds. New York: Plenum Press, 1972, pp. 85–103.

[18] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke, "Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization," *Journal of Computer and System Sciences*, vol. 72, no. 8, pp. 1386–1396, 2006.

[19] S. Thomasse, "A quadratic kernel for feedback vertex set." in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.
[20] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.
[21] J. Scott, T. Ideker, R. Karp, and R. Sharan, "Efficient algorithms for detecting signaling pathways in protein interaction networks," *Journal of Computational Biology*, vol. 13, pp. 133–144, 2006.
[22] S. Bruckner, F. Hüffner, R. Karp, R. Shamir, and R. Sharan, "Topology-free querying of protein interaction networks," in *Proc. 13th Annual International Conference on Computational Molecular Biology (RECOMB), Tucson, USA*. Springer, 2009, p. 74.
[23] H. Bodlaender and A. Koster, "Combinatorial optimization on graphs of bounded treewidth," *The Computer Journal*, 2007.
[24] I. Xenarios, L. Salwinski, X. Duan, P. Higney, S. Kim, and D. Eisenberg, "DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions," *Nucleic Acids Research*, vol. 30, no. 1, p. 303, 2002.
[25] P. Dent, A. Yacoub, P. Fisher, M. Hagan, and S. Grant, "MAPK pathways in radiation responses," *Oncogene*, vol. 22, pp. 5885–5896, 2003.
[26] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori, "The KEGG resource for deciphering the genome," *Nucleic acids research*, vol. 32, pp. 277–280, 2004.
[27] F. Huffner, S. Wernicke, and T. Zichner, "Algorithm Engineering For Color-Coding To Facilitate Signaling Pathway Detection," in *Proceedings of the 5th Asia-Pacific Bioinformatics Conference*. Imperial College Press, 2007.
[28] M. Fellows, "Parameterized complexity: new developments and research frontiers," in *Aspects of Complexity: Minicourses in Algorithmics, Complexity and Computational Algebra: Mathematics Workshop, Kaikoura, January 7-15, 2000*. Walter de Gruyter, 2001, p. 51.
[29] V. Lacroix, C. Fernandes, and M.-F. Sagot, "Motif search in graphs: application to metabolic networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 3, no. 4, pp. 360–368, 2006.
[30] M. Fellows, G. Fertin, D. Hermelin, and S. Vialette, "Sharp tractability borderlines for finding connected motifs in vertex-colored graphs," in *Proc. 34th International Colloquium on Automata, Languages and Programming (ICALP), Wroclaw, Poland*, ser. Lecture Notes in Computer Science, vol. 4596. Springer, 2007, pp. 340–351.
[31] N. Betzler, M. Fellows, C. Komusiewicz, and R. Niedermeier, "Parameterized algorithms and hardness results for some graph motif problems," in *Proc. 19th Annual Symposium on Combinatorial Pattern Matching (CPM), Pisa, Italy*, ser. Lecture Notes in Computer Science, vol. 5029. Springer, 2008, pp. 31–43.
[32] R. Dondi, G. Fertin, and S. Vialette, "Weak pattern matching in colored graphs: Minimizing the number of connected components," in *Proc.10th Italian Conference on Theoretical Computer Science (ICTCS), Roma, Italy*. World-Scientific, 2007, pp. 27–38.
[33] ——, "Maximum motif problem in vertex-colored graphs," in *Proc. 20th Annual Symposium on Combinatorial Pattern Matching (CPM'09), Lille, France*, ser. Lecture Notes in Computer Science, G. Kucherov and E. Ukkonen, Eds., vol. 5577, 2009, pp. 221–235.
[34] G. Blin, F. Sikora, and S. Vialette, "GraMoFoNe: a cytoscape plugin for querying motifs without topology in protein-protein interactions networks," in *2nd International Conference on Bioinformatics and Computational Biology (BICoB-2010)*. International Society for Computers and their Applications (ISCA), 2010.
[35] S. Schbath, V. Lacroix, and M. Sagot, "Assessing the exceptionality of coloured motifs in networks," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2009, 2009.

Guillaume Blin is an associate professor in the LIGM at Université Paris-Est - France. He defended his Ph.D. on november 2005 from Université de Nantes - France. His current research interests include computational complexity and approximation, algorithms, and bioinformatics.

Florian Sikora is a Ph. D. student in the LIGM at Université Paris-Est - France. He received his Master Degree from the Université Paris-Est in 2008. His research interests include algorithmics in computational biology and more especially fixed parameter tractable algorithms.

Stéphane Vialette a Researcher in the LIGM at Université Paris-Est - France. He received his PhD degree in 2001 from the University Paris 7 - France. His research interests are in computational biology, algorithmics and combinatorics.