Querying Protein-Protein Interaction Networks

Guillaume Blin, Florian Sikora, and Stéphane Vialette

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

Abstract. Recent techniques increase the amount of our knowledge of interactions between proteins. To filter, interpret and organize this data, many authors have provided tools for querying patterns in the shape of paths or trees in Protein-Protein Interaction networks. In this paper, we propose an exact algorithm for querying graphs pattern based on dynamic programming and color-coding. We provide an implementation which has been validated on real data.

1 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 has yield to an increase in the interest of proteins (e.g. their numbers, functions, complexity and interactions). Among others protein properties, the set of all their interactions for an organism, called Protein-Protein Interactions (PPI) networks, have recently attracted lot of interest. Knowledge on them increases in an exponential manner due to the use of various genome-scale screening techniques [10,12,23]. Unfortunately, acquiring such valuable resources is prone to high noise rate [10,19].

Comparative analysis of PPI tries to determine the extent to which protein networks are conserved among species. Indeed, numerous evidences suggest that proteins functioning together in a pathway (i.e., a path in the interaction 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 [17].

In this article, we focus on the following related problem called Graph Query (formaly defined later). Given a PPI network and a pattern in the shape of graph, query the pattern in the network consist of find a subnetwork of the PPI network which is most similar as possible to the pattern. Similarity is measured both in terms of sequence similarity and graph topology conservation.

Unfortunately, this problem is clearly equivalent to the NP-complete subgraph homeomorphism problem [9]. Recently, several techniques have been proposed to overcome the difficulty of this problem. By restricting the query to a path, Kelley *et al.* [16] were able 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 $f(k)n^{O(1)}$ time where f is a function only depending on the parameter k, and n is the size of the input [8]. Pinter *et al.* [18] proposed an algorithm dealing with tree shape query that is restricted to forest PPI networks (i.e., collection of trees).

Later on, Shlomi *et al.* [21] proposed an alternative, called QPath [16], for querying paths in a PPI network which is based on the color-coding technique introduced by Alon, Yuster and Zwick [1]. In addiction of being faster, QPath allows more flexibility by considering non-exact matches. Finally, Dost *et al.* [7] developed QNet, an algorithm to handle tree query in the general context of PPI networks. The authors also gave some theoretical approaches for querying graphs by using the tree decomposition of the query.

Since QNet is the major reference in this field and is quite related to the work presented in this article, let us present it briefly. QNet is an exact FPT algorithm for querying trees in a PPI network. The complexity is $2^{O(k)}m$, where k is the number of proteins in the query and m the number of edges of the PPI network. As QPath, QNet uses dynamic programming and color-coding. For querying graphs in a network, QNet uses, as a subroutine, an exact algorithm to query trees. To do so, they perform a tree decomposition. A formal definition of a tree decomposition can be found in [4]. Roughly speaking, it is a transformation of a graph into a tree. A tree node (or a bag) can contain several graph nodes. There are several ways 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 this treewidth is NP-Hard [3]. From this tree decomposition, the time complexity of QNet is $O(2^{O(k)}n^{t+1})$ time, where k is the size of the query, n is the size of the PPI network, and t is the treewidth of the query.

QNet is an exact algorithm for querying trees in a PPI network. A logical extension would be to query graphs. The authors of [7] provides a theoretical solution, without implementation and which depends on the query treewidth. We propose in this article an exact alternative solution, using color-coding (Section 2). We provide in Section 3 some experimental results.

2 PADA1 as an alternative to QNet

In this section, we propose an alternative to QNet called PADA1 (Protein Alignment Dealing with grAphs). At the broadest 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). It is worth mentioning that, following 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 an alternative to QNet.

2.1 Transforming the query into a tree

We begin by presenting Algorithm Graph2Tree to transform a graph $G = (V, E)$ into a tree, without loss of information (*i.e.*, one can reconstruct the graph starting from the tree). Informally, 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 one edge deletion. Central is our approach is thus the node duplication procedure (Algorithm DUPLICATE), see Figure 1 for an illustration to break a cycle at vertex v_1. For each $u \in V$, write $d(u)$ for the set of all copies of vertex u including itself.

```
Function GRAPH2TREE($G$)
begin
    $d(u) \leftarrow u$ for all $u$ of $V$;
    for ($i = 0$ ; $i < |V|$ ; $i++$) do
        foreach subgraph $G' = (V', E')$ of $G$ such that $|V'| = |V| - i$ do
            if $G'$ is acyclic then
                foreach node $u$ of $V \backslash V'$ do
                    foreach $(u, v) \in E$ do
                        $tmp \leftarrow G$;
                        DUPLICATE($v$, $u$, $d$);
                        if $G$ is not connected anymore then
                            $G \leftarrow tmp$;
                        end
                    end
                end
                return $G$;
            end
        end
    end
end
```

Algorithm 1: "*Brute-force*" transformation algorithm

```
Function DUPLICATE($G = (V, E)$, $v_a$, $v_b$, $d$)
begin
    Let $i \leftarrow |d(v_b)|$;
    $V \leftarrow V \cup \{v_{b\_i}\}$;
    $d(v_b) \leftarrow d(v_b) \cup \{v_{b\_i}\}$ ;
    $E \leftarrow E - \{(v_a, v_b)\}$;
    $E \leftarrow E \cup \{(v_a, v_{b\_i})\}$;
end
```

Algorithm 2: Algorithm to duplicate a node when a cycle is detected.

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

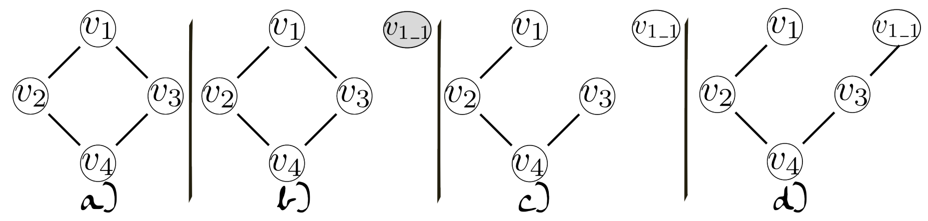

Fig. 1. Steps when Duplicate(G, v_3, v_1, d) is called on graph a). b) A node v_{1_1} from v_1 is created. c) The edge (v_3,v_1) is deleted: the cycle is then broken. d) The edge (v_3,v_{1_1}) is added. Finally, the resulting graph is acyclic, and $d(v_1) = \{v_1, v_{1\_1}\}$.

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 overall number of duplications. Minimizing the cardinality of F is the well-known NP-complete Feedback Vertex Set problem [15]: Given a graph G, find a minimum cardinality subset of vertices with the property that removal of these vertices from the graph eliminates all cycles.

We only have implemented an algorithm using a "brute-force" solution for the Feedback Vertex Set problem. Since there are $2^{|V|}$ potential subgraphs, its complexity is $O(2^{|V|} \times |E|)$, but it is still running in seconds. 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.* [11], using *iterative compression*, or a cubic [5] or quadratic kernalization [22].

2.2 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.

Let us fix notations. PPI networks are represented by undirected 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. There is no weight for these later.

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 BLAST [2]). 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 common 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 $\sigma : V_Q \rightarrow V_A \cup \{\texttt{del}\}$. More precisely, the function σ is defined such that for all q of V_Q, $\sigma(q) = v$ if and only if q and v are homologous, and $\sigma(q) = \texttt{del}$ otherwise.

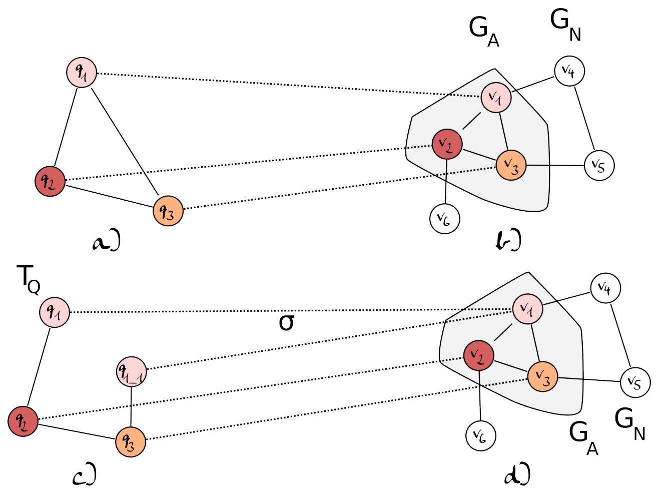

Fig. 2. a) The graph query with a cycle, before calling Graph2Tree algorithm. c) The query after calling Graph2Tree where q_1 has been duplicated. Thus, q_1 and q_{1_1} have to be aligned with the same node of the network. b) and d) denote the resulting graph alignment G_A, subgraph of the network G_N. The horizontal dashed lines denote a match between two proteins.

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. 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, a PPI network G_N, a similarity function h, penalty scores δ_i and δ_d for each insertion and deletion, find an alignment (G_A, σ) between T_Q and G_N of maximal 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 all edges involved in G_A (*i.e.*, $\sum_{e \in E_A} w(e)$), (iii) a penalty score δ_d for each node deletion (*i.e.*, $\sum_{\substack{q \in V_Q \\ \sigma(q) = \texttt{del}}} \delta_d$), 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 [1] and dynamic programming. This randomized technique allows to find simple paths of length k in a network in $O(2^k)$ time (instead of the brute-force $O(n^k)$ time algorithm), where n is the number of proteins in the network [20]. In [7], 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 $O(n^k)$ potential alignments, with $n = |V_N|$ and $k = |V_Q|$.

Using color-coding, one may decrease this complexity to $O(2^k)$. 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 potential solution (*i.e.*, 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 $O(2^k)$. In order to get an optimal solution, this process is repeated. More precisely, according to QNet [7], since a colorful alignment happens with probability $\frac{k!}{k^k} \simeq e^{-k}$, the coloration step has to be done $\log(\frac{1}{\epsilon})e^k$ 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) of E_Q when one considers q_i of V_Q aligned with a node v_i of 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, (iii) a deletion of q_j. This is made 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). To deal with duplicated nodes (cf. Graph2Tree algorithm), we pre-compute all possible assignment of the duplicated nodes V_Q of T_Q. More precisely, for each q of F and for all q' of $d(q)$ one assign $\sigma(q')$ with each v of V_N. We then compute for each assignment A an alignment with respect to A. We denote BestConstraintAlignment this step (details omitted due to space constraints). The difficulty is to construct 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 the 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 multi-set 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.

Algorithm 3 may be summarized as follow. Perform $\log(\frac{1}{\epsilon})e^k$ random colorations of the PPI network G_N to ensure optimality with a probability of at least $1 - \epsilon$. A coloration consists in affecting a random color of S_C to each node of V_N. Then, for each coloration, we build all possible valid assignments A of the

```
1  Function PADA1 ($T_Q$,$G_N$, $h$, $threshold$)
2  begin
3      $BestG_A \leftarrow \emptyset$; $BestScore \leftarrow -\infty$;
4      for ($i = 0$; $i < \log(\frac{1}{\epsilon})e^k$; $i++$) do
5          randomly colorize $G_N$ with $k + N_{ins}$ colors;
6          foreach valid assignment $A$ do
7              $G_A \leftarrow$ BestConstraintAlignment($G_N$, $T_Q$, $A$, $h$, $threshold$);
8              if $score(G_A) > BestScore$ then
9                  $BestG_A \leftarrow G_A$;
10                 $BestScore \leftarrow score(G_A)$;
11             end
12         end
13     end
14     return $BestG_A$;
15 end
```

Algorithm 3: Sketch of the PADA1 algorithm to align a query graph to a network.

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 alignment according to A with Algorithm BestConstraintAlignment. We keep the best score of these trials, and, get the corresponding alignment by classic backtracking technique.

Let us 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 $\log(\frac{1}{\epsilon})e^k$ iterations). The random coloration can be done in $O(n)$, where $n = |V_N|$. There are $n^{|F|}$ possible assignments. The complexity of BestConstraintAlignment is $2^{O(k+N_{ins})}m$ as in QNet, where k is the size of the graph query and $m = |E_N|$, since our modifications are essentially additional tests which are 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 $O(n^{|F|}.2^{O(k+N_{ins})}m)$ time. Observe that the time complexity does not depends on the total number of duplicated nodes but on the size of F.

3 Experimental results

According to the authors of QNet, one may query a PPI network by running a $O(2^{O(k)}n^{t+1})$ time algorithm $\log(\frac{1}{\epsilon})e^k$ times, where t is the treewidth of the query. Thus, the difference between the two algorithms is mainly related to $t+1$ versus $|F|$ question (*i.e.*, the size of the set of families of duplicated nodes computed by Algorithm Graph2Tree). These two parameters are not easily comparable, except for trivial cases. However, we have computed some experimental tests to compare these two parameters on random graphs. Figure 3 suggests that parameter $|F|$ is usually smaller for moderate size graphs (*i.e.*, those graphs for which PADA1 is still practicable). Observe however that there are graphs with treewidth smaller than $|F|$, and hence no definitive conclusion can be drawn.

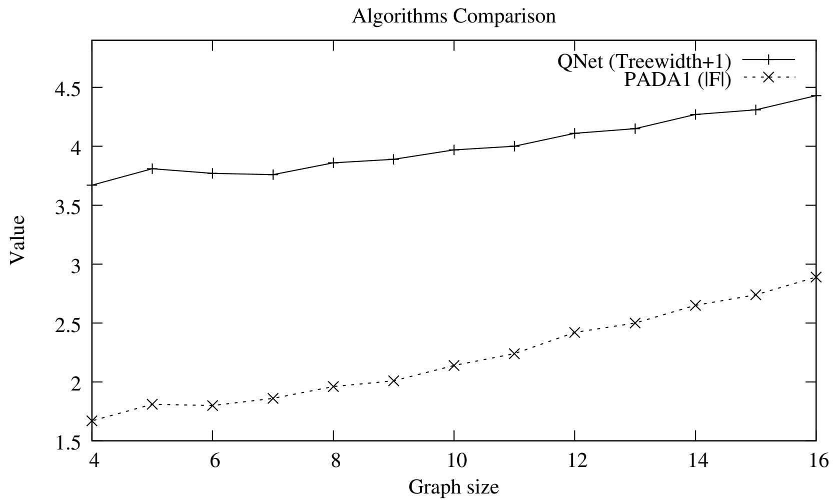

Fig. 3. Comparison between QNet (*i.e.*, the treewidth+1 value) and PADA1 (i.e., the size of F computed after running the GRAPH2TREE algorithm). The method is as follows: for each different size of graph, we get the average treewidth and F values over 30 000 connected graphs, randomly constructed with the NetworkX library (http://networkx.lanl.gov/). Treewidth is computed with the exact algorithm provided by http://www.treewidth.com/, while the size of F is computed with our GRAPH2TREE 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. So, there are not n possibilities for each element of F. The number of executions 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 less than the worst case time complexity. Therefore, and not surprisingly, the BLAST threshold used to determine if a protein is homologous to another have 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 cases.

In order to validate our algorithm, we perform the experimental tests on real data, proposed by QNet [7]. 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 [6], 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 [14] 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. $|F|$. Moreover, we probably could certainly speed-up the running time by using the Hüffner *et al.* technique [13], which basically consists in increasing the number of colors used during the coloration step.

4 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. The time complexity of this algorithm is $n^{|F|}2^{O(k)}$, where n is the size of the PPI network, k is the size of the query, 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 is the main difference with QNet of Dost *et al.* [7], which uses the treewidth of the query (unimplemented algorithm). We have performed some tests on real data and have

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

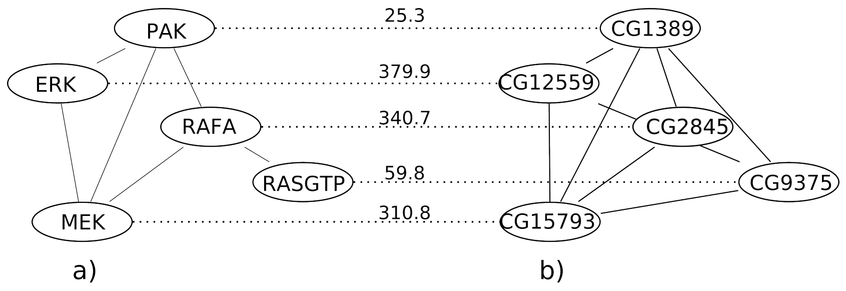

Fig. 4. A result sample of our algorithm. a) A MAPK human query, get from [14], with three cycles. b) The alignment graph given by our algorithm in the fly PPI network. Dashed lines denotes the BLAST homology scores between the two proteins. Our algorithm retrieves a query graph in an other network. As in QNet [7], it seems to be that there is some conservation between these two species.

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, depends on the initial topology of the query graph. Obtaining more information about the topology of the queries is of particular interest in this context. Future works includes using this information to predict average time complexity.

References

1. N. Alon, R. Yuster, and U. Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, 1995.
2. S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.
3. S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.
4. H.L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.
5. H.L. Bodlaender. A cubic kernel for feedback vertex set. In *Proceedings 24th International Symposium on Theoretical Aspects of Computer Sciences, STACS*, pages 320–331, 2007.
6. P. Dent, A. Yacoub, P.B. Fisher, M.P. Hagan, and S. Grant. MAPK pathways in radiation responses. *Oncogene*, 22:5885–5896, 2003.
7. B. Dost, T. Shlomi, N. Gupta, E. Ruppin, V. Bafna, and R. Sharan. QNet: A Tool for Querying Protein Interaction Networks. *RECOMB*, pages 1–15, 2007.
8. R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
9. M.R. Garey and D.S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W.H. Freeman, San Franciso, 1979.
10. A.C. Gavin, M. Boshe, et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 414(6868):141–147, 2002.

11. 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*, 72(8):1386–1396, 2006.
12. Y. Ho, A. Gruhler, et al. Systematic identification of protein complexes in Saccharomyces cerevisae by mass spectrometry. *Nature*, 415(6868):180–183, 2002.
13. 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.
14. M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resource for deciphering the genome. *Nucleic acids research*, 32:277–280, 2004.
15. R.M. Karp. Reducibility among combinatorial problems. In J.W. Thatcher and R.E Miller, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.
16. B.P. Kelley, R. Sharan, R.M. Karp, T. Sittler, D. E. Root, B.R. Stockwell, and T. Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences*, 100(20):11394–11399, 2003.
17. M. Pellegrini, E.M. Marcotte, M.J. Thompson, D. Eisenberg, and T.O. Yeates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *PNAS*, 96(8):4285–4288, 1999.
18. R.Y. Pinter, O. Rokhlenko, E. Yeger-Lotem, and M. Ziv-Ukelson. Alignment of metabolic pathways. *Bioinformatics*, 21(16):3401–3408, 2005.
19. T. Reguly, A. Breitkreutz, L. Boucher, B.J. Breitkreutz, G.C. Hon, C.L. Myers, A. Parsons, H. Friesen, R. Oughtred, A. Tong, et al. Comprehensive curation and analysis of global interaction networks in saccharomyces cerevisiae. *Journal of Biology*, 2006.
20. J. Scott, T. Ideker, R.M. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13:133–144, 2006.
21. T. Shlomi, D. Segal, E. Ruppin, and R. Sharan. QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, 7:199, 2006.
22. S. Thomasse. A quadratic kernel for feedback vertex set. In *Unpublished manuscript. To appear in Proceedings SODA*, 2009.
23. P. Uetz, L. Giot, et al. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisae. *Nature*, 403(6770):623–627, 2000.
24. I. Xenarios, L. Salwinski, X.J. Duan, P. Higney, S.M. Kim, and D. Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30(1):303, 2002.

Appendix

In the following,

- S represents the multi-set of colors,
- A represents the pre-computed assignment for duplicated nodes,
- $G_N = (V_N, E_N)$ denotes the PPI network,
- $T_Q = (V_Q, E_Q)$ denotes the query after calling Graph2Tree,
- *root* is an arbitrary root of T_Q,
- $v \in V_N$,
- $q \in V_Q$,
- q_j denotes the j^{th} child of q,
- n_{q_j} denotes the number of child nodes of q,
- c is a function which gives the color of a node $v \in V_N$.

Thereafter, we give technical details to compute BestConstraintAlignment, resulting in G_A, according to the pre-computed assignment A.

$G_A \leftarrow \max_v W^M(root, v, S, 1, A)$
If $|S| \leq 1$, $W^M(q, v, S, j, A) \leftarrow -\infty$
Else,

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

To compute this value, we consider q as aligned with v. Then, we perform the best action for q_j (i.e., a match with a neighbor u of v, an insertion of a neighbor u of v, a deletion of q_j).

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

If q is not a duplicated node, the value is the similarity score with v given by h. Otherwise, the assignment of q has been done in A, and has to be preserved.

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

The node v is consider to be inserted. One has to restart the alignment from a neighbor u of v. If v has to be aligned with a duplicate node (i.e., $A^{-1}(v) \neq \emptyset$), inserting v is forbidden. Indeed, if it is allowed, v will be inserted and the alignment of v with $A^{-1}(v)$ will be impossible.

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

The node v and the father of q are aligned. The node q is considered to be deleted. The alignment restart from q_1, the unique son of q. If q is a duplicated node, the deletion is only allowed when $A(q) = \textit{deletion}$.