Algorithms for subnetwork mining in heterogeneous networks

Guillaume Fertin, Hafedh Mohamed-Babou, Irena Rusu

Algorithms for subnetwork mining in heterogeneous networks

Guillaume Fertin, Hafedh Mohamed-Babou and Irena Rusu

LINA, UMR 6241, Université de Nantes, France
{Guillaume.Fertin, Hafedh.Mohamed-Babou, Irena.Rusu}@univ-nantes.fr

Abstract. Subnetwork mining is an essential issue in network analysis, with specific applications *e.g.* in biological networks, social networks, information networks and communication networks. Recent applications require the extraction of subnetworks (or *patterns*) involving several relations between the objects of interest, each such relation being given as a network. The complexity of a particular mining problem increases with the different nature of the networks, their number, their size, the topology of the requested pattern, the criteria to optimize. In this emerging field, our paper deals with two networks respectively represented as a directed acyclic graph and an undirected graph, on the same vertex set. The sought pattern is a longest path in the directed graph whose vertex set induces a connected subgraph in the undirected graph. This problem has immediate applications in biological networks, and predictable applications in social, information and communication networks. We study the complexity of the problem, thus identifying polynomial, NP-complete and APX-hard cases. In order to solve the difficult cases, we propose a heuristic and a branch-and-bound algorithm. We further perform experimental evaluation on both simulated and real data.

1 Introduction

The use of communication, social and telecommunication networks has dramatically increased recently, resulting in new prominent applications of network analysis. In addition to these real-world applications, network representations of new types of data - and particularly biological data - highlight the drastic need for a new, multi-dimensional, type of (sub)network mining in which several networks, representing several relations between the same objects, are simultaneously investigated for the extraction of a multi-dimensional pattern [5,13,15].

The study of multi-dimensional mining started several years ago, but it mainly concerns homogeneous representations of data: directed graph alignment [4], undirected graph alignment [6], relational data mining [8], social networks mining [13] are several examples. Recently, such approaches found applications in computational biology [12,14,16], but also

showed their limits, due to the multiple types of biological networks that are used to describe different views of the same biological process. In such applications, a process is often represented as a path in a directed network (e.g., a metabolic network), and as a connected graph in an undirected network (e.g., a protein-protein interaction network); the link between the two networks is then ensured by the components involved in the process, that are represented as vertices in each network. Identifying a particular biological process then requires to identify parts of the two networks (directed and undirected) that have the suited topological patterns and the same vertex set. The need for such applications to replace either manual or case-by-case studies is nowadays fundamental [3,7,15,17].

In this paper, we approach multi-dimensional mining within two heterogeneous networks, driven by the previously cited applications in biological networks. The paper is organized as follows. Section 2 presents the problem. In Section 3, we show that the problem is APX-hard in the general case, NP-complete even in restricted cases, and exhibit classes of instances for which the problem is polynomial. In Section 4, we propose a heuristic and a branch-and-bound algorithm, that we evaluate in Section 5 both on simulated and real (biological) data. Section 6 is the conclusion. Note that due to space constraints, some proofs and illustrations are omitted.

2 The problem

All along the paper, D will denote a directed graph and G an undirected graph, built on the same set of vertices V. In general, given a graph H, $V(H)$ is its vertex set. If H is undirected (resp. directed) then its edge set (resp. its arc set) is denoted $E(H)$ (resp. $A(H)$). Given a set $S \subseteq V(H)$, we denote $H[S]$ the subgraph of H induced by S.

A (D, G)-*consistent path* is a (directed) path P in D such that $G[V(P)]$ is connected. The Skew SubGraph Mining problem (abbreviated SkewGraM) is formulated as follows:

SkewGraM
Instance : A DAG D and an undirected graph G.
Requires : Find a longest (D, G)-consistent path.

Several more general variants of the problem (*e.g.* when D has circuits, or when D and G have different, but related, vertex sets) may be useful in practice, but they should be reduced to this simpler variant, for which we provide effective solutions. See [2] for details.

3 The complexity of SkewGraM

We study the complexity of SkewGraM considering different topological constraints on graphs D and G. In Table 1, D^* is the *underlying graph* of D, obtained by removing the arc orientations. A *star* (resp. *bi-star*) is a tree whose number of vertices with degree 2 or more is exactly one (resp. two), whereas an *outerplanar graph* is a graph admitting a planar embedding with all vertices on a circle, all edges inside the circle and such that edges do not cross each other. Recall that the *diameter* of a graph is the maximum length of a shortest path between any two of its vertices.

G \ D^*	Tree	Outerplanar	General graph
Chordless path or cycle, (bi-)star	P [Lem. 1]		
Tree with diameter 4	P [Lem. 1]	NPC [Thm. 1]	NPC [Thm. 1]
General graph	P [Lem. 1]	NPC [Thm. 1]	NPC [Thm. 1] APX-hard [Thm. 2]

Table 1. Complexity of the SkewGraM problem.

According to Table 1, SkewGraM is polynomially solvable as soon as D^* is a tree, but it becomes NP-complete even for relatively simple graph structures, *e.g.* when G is a tree with diameter 4 and D^* is an outerplanar graph. In the most difficult cases, the problem is APX-hard.

The following lemma gathers together the polynomial-time solvable cases we identified:

Lemma 1. SkewGraM *is polynomial-time solvable when at least one of the following conditions holds: a) D^* is a tree. b) G is a chordless path or cycle. c) G is a (bi-)star.*

The theorem below shows the NP-completeness of the problem in a particular configuration.

Theorem 1. SkewGraM *(in its decision version) is* NP*-complete, even when D^* is an outerplanar graph and G is a tree with diameter 4.*

Proof. The problem is clearly in NP. We propose a reduction from MAX 2Sat[11]. Let $\mathcal{C} = \{C_1, \ldots, C_p\}$ be a collection of p clauses with two literals each, over the variable set $\mathcal{X}_n = \{x_1, \ldots x_n\}$. Let D be built on $2p+2n+2$ levels, called *optional* (marked with a star) or *compulsory* (not marked):

- level 0 : a vertex s;

- level* $2i-1$, $1 \leq i \leq p$: two vertices $v_{i,1}$ and $v_{i,2}$ corresponding to the literals of clause C_i;
- level $2i$, $1 \leq i \leq p$: a vertex c_i corresponding to the clause C_i;
- level $2p+1$: two vertices $v_{p+1,1}$ and $v_{p+1,2}$ corresponding, respectively, to variables x_n and $\overline{x_n}$;
- level $2p+2$: a vertex c_{p+1};
- level $2p+2+2i-1$, $1 \leq i \leq n$: two vertices a_i and b_i;
- level $2p+2+2i$, $1 \leq i < n$: a vertex A_i.

Then add (a) all possible arcs between any two consecutive levels, (b) the arc sc_1 and (c) the arcs $c_ic_{i+1}, 1 \leq i < p$. It is clear that D is a DAG. To see that D^* is an outerplanar graph, it is sufficient to draw the vertices on a circle according to the order $s, v_{1,1}, c_1, v_{2,1}, c_2, \ldots, v_{p+1,1}$, c_{p+1}, a_1, $A_1, \ldots, a_{n-1}$, $A_{n-1}, a_n, b_n, \ldots, b_1, v_{p+1,2}, \ldots, v_{1,2}$.

Graph G is a tree with root s. There is an edge between s and each vertex in $\{a_i, b_i : 1 \leq i \leq n\} \cup \{A_i : 1 \leq i < n\} \cup \{c_i : 1 \leq i \leq p+1\}$. There is an edge between each vertex a_i (resp. b_i) and any vertex $v_{l,m}$ with $1 \leq l \leq p+1, 1 \leq m \leq 2$, such that $v_{l,m}$ corresponds to the literal x_i (resp. $\overline{x_i}$). Obviously, G has diameter 4. We claim that there is an assignment for the variables in $\mathcal{X}_n$ that satisfies at least k clauses if and only if there is a (D, G)-consistent path with length at least $p+k+1+2n$. This assertion is based on the following remarks:

1. Each consistent path of length at least 1 contains s. Indeed, the connected components of $G - \{s\}$ do not allow to compute paths in D.
2. No consistent path P contains two vertices associated with literals $x_i, \overline{x_i}$, for some i. Indeed, at most one of the vertices a_i, b_i belongs to P (by construction of D), and thus in G only vertices corresponding to x_i or to $\overline{x_i}$ may be connected to s (via a_i or, respectively, b_i).
3. Every consistent path P of length at least $p+1$ necessarily contains one vertex from each compulsory level. Indeed, such a path contains s (as before) and, because of its length, at least one vertex $v_{i,j}$, $1 \leq i \leq p+1$ and $1 \leq j \leq 2$. The connectivity in G implies that a_i or b_i belongs to P and consequently, in D, we deduce that $v_{p+1,1}$ or $v_{p+1,2}$ (that correspond respectively to x_n and $\overline{x_n}$) belongs to P. Then, the connectivity in G implies that a_n or b_n belongs to P and the claim follows.

$\Rightarrow$: Given an assignment $\mathcal{A}$ of the variable set $\mathcal{X}_n$ that satisfies k' clauses of $\mathcal{C}$, s.t. $k' \geq k$, assume w.l.o.g. that variables $v_{i_1,1}, \ldots, v_{i_{k'},1}$, $v_{p+1,1}$ correspond to true literals. Let $\mathcal{B}(i) = a_i$ if x_i is true, and $\mathcal{B}(i) = b_i$ otherwise. Then the path P with vertices $s, v_{i_1,1}, \ldots, v_{i_{k'},1}, v_{p+1,1}, c_1, \ldots,$

$c_{p+1}, \mathcal{B}(1), A_1, \mathcal{B}(2), A_2, \ldots, \mathcal{B}(n-1), A_{n-1}, \mathcal{B}(n)$ is (D, G)-consistent and has length $p + k' + 1 + 2n$. Indeed, in G the vertices $v_{i_1,1}, \ldots, v_{i_{k'},1}, v_{p+1,1}$ are connected to s using the corresponding vertex $\mathcal{B}(l_{i_j})$ (s. t. $v_{i_j,1}$ is $x_{l_{i_j}}$ or $\overline{x_{l_{i_j}}}$), $1 \leq j \leq k'$, and $\mathcal{B}(n)$ respectively. All the other vertices are adjacent to s.
$\Leftarrow$: Let P be a (D, G)-consistent path P s.t. $|V(P)| \geq p+k+2+2n$. Let k' be the number of vertices in P that belong to optional levels. According to Remark 2, assigning the value `true` to the literals associated to these vertices yields a correct assignment, that satisfies k' clauses. Moreover, by Remarks 1 and 3, P contains $p+2+2n$ vertices on the compulsory levels, and thus $|V(P)| = p + 2 + k' + 2n$ vertices. We deduce that $k' \geq k$. □

Moreover, we can also show (proof omitted here) using a reduction from Maximum Independent Set on cubic graphs that:

Theorem 2. SkewGraM *is* APX*-hard.*

4 Two algorithms for SkewGraM

We propose two algorithms for SkewGraM: a heuristic called `AlgoH`, and an exact exponential-time algorithm called `AlgoBB` using the branch and bound method. Both algorithms look for a longest (D, G)-consistent path going through a given arc xy of D. Then, to solve SkewGraM, an execution is needed for every arc xy of D.

Let $i \rightsquigarrow^D j$ denote a path in D from vertex i to vertex j (becomes $i \rightarrow^D j$ when reduced to an edge). Given two undirected graphs $G_1(U, E_1)$ and $G_2(U, E_2)$, a *common connected component* of G_1 and G_2 is any maximal set $X \subseteq U$ such that $G_1[X]$ and $G_2[X]$ are connected. We note by $CCC(D^*, G, i \rightsquigarrow^D j)$ the common connected component of D^* and G that contains all the vertices of the path $i \rightsquigarrow^D j$, if such a common connected component exists (equal to $\emptyset$ otherwise). The notation S_i^+ stands for the set of vertices that are reachable by a path from vertex i in D, whereas S_i^- stands for the set of vertices of D reaching vertex i by a path in D. Finally, vertex $r \in V$ is called a *bridge of* $i \rightsquigarrow^D j$ *with respect to* G if there is no common connected component of $D^*[V - \{r\}]$ and $G[V - \{r\}]$ containing all the vertices of $i \rightsquigarrow^D j$ (i.e. $CCC(D^*[V-\{r\}], G[V-\{r\}], i \rightsquigarrow^D j) = \emptyset$).

The heuristic `AlgoH`. We construct the (D,G)-consistent path progressively by starting with the given arc xy and extending it. `AlgoH` first computes the cover set of a path, which is used to reduce the graph by removing vertices not compatible with the bridges.

Algorithm 1 GetCoverSet$(D, G, i \rightsquigarrow^D j)$

Require: A DAG $D = (V, A(D))$, an undirected graph $G(V, E(G))$, a path $i \rightsquigarrow^D j$.
Ensure: Computes the cover set of $i \rightsquigarrow^D j$.

```
1: S := S_i^- ∪ S_j^+ ∪ V(i ⇝^D j)
2: S := CCC(D*[S], G[S], i ⇝^{D[S]} j); STOP := false;
3: while ((STOP = false) and (S ≠ ∅)) do
4:     S_tmp := S; /* note that i ⇝^{D[S]} j is identical to i ⇝^D j */
5:     for each bridge r of i ⇝^D j in G do
6:         S_tmp := S_tmp ∩ ({r} ∪ S_r^- ∪ S_r^+)
7:     end for
8:     if (S = S_tmp) then
9:         STOP := true
10:    else
11:        S := S_tmp; S := CCC(D*[S], G[S], i ⇝^{D[S]} j);
12:    end if
13: end while
14: return S
```

Definition 1. *The* cover set *of a path $i \rightsquigarrow^D j$, denoted* CoverSet*(D,G, $i \rightsquigarrow^D j$), is the set X satisfying:*

1. *$V(i \rightsquigarrow^D j) \subseteq X \subseteq S_i^- \cup S_j^+ \cup V(i \rightsquigarrow^D j)$.*
2. *$D^*[X]$ and $G[X]$ are connected.*
3. *If r is a bridge of $i \rightsquigarrow^{D[X]} j$ w.r.t. $G[X]$ then $X \subseteq S_r^- \cup S_r^+ \cup \{r\}$.*
4. *X is maximal (with respect to the inclusion order).*

If, for a path $i \rightsquigarrow^D j$, no vertex set X satisfies conditions 1., 2. *and* 3., *then by convention* CoverSet*(D,G, $i \rightsquigarrow^D j$)$= \emptyset$.

The cover set of a path is unique, and easily computable (see Algorithm 1 which uses Algorithm GenPartRefinement described in [10] to compute the common connected components):

Lemma 2. *The cover set of a given path $i \rightsquigarrow^D j$ is well-defined.*

Lemma 3. *Algorithm* GetCoverSet *correctly computes the cover set of a given path in $\mathcal{O}(n^2 \log n + nm \log^2 n)$.*

Now, to compute a (D, G)-consistent path going through xy, we use Algorithm AlgoH (see Algorithm 2) to successively increase the current path cp (which is initially the arc $x \rightarrow^D y$) as follows. Once Algorithm GetCoverSet is applied to reduce D and G when possible (line 7), either $V(D) = \emptyset$ (and there is no (D, G)-consistent path containing xy), or D is Hamiltonian (and the algorithm returns the best current solution), or cp must be extended. The possible extensions p (by adding one vertex at the beginning or at the end of the path, using function Extend) are computed (lines 12-16) as well as the resulting reduced graphs (line 17).

Algorithm 2 ALGOH(D, G, xy)

Require: A DAG $D = (V, A(D))$, an undirected graph $G(V, E(G))$, an arc $xy \in A(D)$.
Ensure: $S \subseteq V$: $D[S]$ is a (D, G)-consistent path containing the arc xy or $S = \emptyset$.

```
 1: /* bcs: best current solution; cp: current path */
 2: /* L_ext: the list of all paths in D extending the current path by one vertex */
 3: /* DCS: the subgraphs of D induced by the cover sets of paths in L_ext */
 4: /* HCS: the Hamiltonian subgraphs induced by the cover sets of paths in L_ext */
 5: bcs := ∅; cp := x →^D y; f := x; l := y; STOP := false;
 6: while (STOP = false) do
 7:    S := GetCoverSet(D, G, cp); D := D[S]; G := G[S];
 8:    if (S = ∅ or D is Hamiltonian) then
 9:       STOP := true;
10:       if |bcs| > |S| then S := bcs end if
11:    else
12:       L_ext := ∅; /* compute the extensions of cp = f ⇝^D l */
13:       for each v that is a predecessor of f or a successor of l do
14:          p = Extend(cp, v); /* extends cp with v */
15:          L_ext := L_ext ∪ {p};
16:       end for
17:       DCS = {D[CoverSet(D, G, p)] : p ∈ L_ext};
18:       HCS := {d ∈ DCS : d is Hamiltonian};
19:       Let h_max ∈ HCS s.t. |V(h_max)| = max{|V(h)| : h ∈ HCS}
20:       if |bcs| < |V(h_max)| then bcs := V(h_max) end if
21:       Let p_max = f_max ⇝^D l_max s.t. value(p_max) = max{value(p) : p ∈ L_ext};
22:       if |bcs| ≥ value(p_max) then
23:          /* No (D, G)-consistent path through xy and longer than |bcs| exists */
24:          S := bcs; STOP := true;
25:       else
26:          cp := p_max; /* continue with the most promising extension */
27:          f := f_max; l := l_max
28:       end if
29:    end if
30: end while
31: return S
```

The Hamiltonian graphs among them are considered for improving the best current value (lines 18-20). Then, the most promising extension is computed using the evaluation $value(p)$, equal to the length of the longest path in $D[\text{CoverSet}(D, G, p)]$ (line 21). If this extension allows to hope an improvement of *bcs*, then it is kept (lines 22-28). An experimental evaluation of ALGOH is given in the next section.

Complexity of Algorithm ALGOH: let Δ be the maximum total degree of a vertex in D, and L be the length of the optimal solution of SkewGraM. Then, the **while** loop in line 6 is executed at most L times. The most time consuming internal instruction is the one in line 17, which makes 2Δ calls to Algorithm GETCOVERSET. Recall that the longest path is easily computed in a DAG. Then the complexity of Algorithm ALGOH is in $\mathcal{O}(\Delta L(n^2 \log n + nm \log^2 n))$.

The exact algorithm `AlgoBB`**.** This algorithm is based on the branch and bound method. The tree TS of sub-solutions is built as follows. The root is associated to the arc xy given as input of SkewGraM. Each vertex s of TS is associated to a path $p(s)$ extending the arc xy. At the end of the construction of TS, its leaves are associated to (D, G)-consistent paths containing xy. The solution of SkewGraM is thus a longest path $i \rightsquigarrow^D j$ such that there exists a leaf of TS associated to $i \rightsquigarrow^D j$.
Branching. We expand vertex s with $p(s) = v_l \rightsquigarrow^D v_m$ as follows. For each v_k that is a predecessor of v_l (resp. successor of v_m), we add in TS a child of s associated to the path $v_k.p(s)$ (resp. $p(s).v_k$). For a vertex $s \in TS$, recall that $value(p(s))$ is the length of the longest path in $D[\textsc{CoverSet}(D, G, p(s))]$. Let $BBvalue(s)$ denote the evaluation of a vertex s in TS. This function is defined as follows: (i) if s is to be expanded, then $BBvalue(s) = value(p(s))$; (ii) if s has already been expanded at some specific moment of the construction of TS, then $BBvalue(s)$ is the length of $p(s)$ if it is (D, G)-consistent. Otherwise, $BBvalue(s) = 0$. Using this evaluation function, we define the *bounding* and *pruning* rules as follows.
*Bounding (**Rule 1**).* Among vertices $\{s_1, s_2, \ldots, s_k\}$ to be expanded, we choose the vertex s^* such that $BBvalue(s^*) = \max\{BBvalue(s_i) : 1 \leq i \leq k\}$. If there are several such vertices, we arbitrarily choose one.
*Pruning (**Rule 2**).* Let s_{max} be a vertex of TS satisfying the two following conditions: (i) s_{max} was expanded, and (ii) $BBvalue(s_{max}) \geq BBvalue(s)$, for any expanded vertex s of TS. Then, delete from TS any leaf vertex s with $BBvalue(s) \leq BBvalue(s_{max})$. This deletion is applied recursively for vertices that become leaves after the deletion of all their children.

Theorem 3. *Algorithm* `AlgoBB` *exactly solves* SkewGraM.

5 Experimental results

In order to show the reliability of our heuristic `AlgoH`, we first applied it on random (Erdös-Rényi and scale-free) graphs and we compared the obtained solutions to the optimal ones computed by our exact algorithm `AlgoBB`. We also applied `AlgoH` on different types of biological networks.

5.1 Performances of `AlgoH`

Let $|\texttt{AlgoBB}|$ (resp. $|\texttt{AlgoH}|$) be the number of vertices of a solution found by `AlgoBB` (resp. found by `AlgoH`). We measured the performance of `AlgoH`

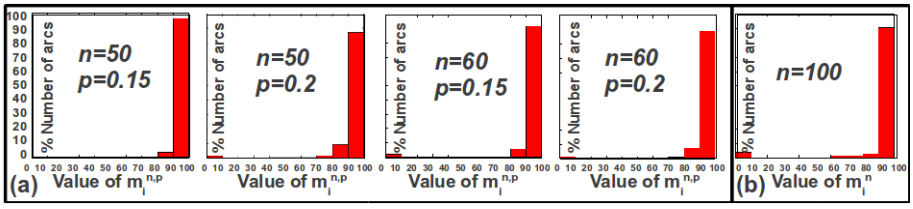

Fig. 1. Performances of heuristic `AlgoH`. We show the percentage of arcs whose $\frac{|\text{ALGOH}|}{|\text{ALGOBB}|} \times 100$ belongs to an interval I_i (see Section 5.2). **(a) General graphs**. For fixed n and p we generated 100 couples (D, G). **(b) Scale-free graphs**. Algorithms `AlgoH` and `AlgoBB` were run on 100 couples (D, G) of order 100.

by computing the ratio $\rho = \frac{|\text{ALGOH}|}{|\text{ALGOBB}|}$ for every input instance. By convention, $\rho = 1$ whenever the exact algorithm finds no (D, G)-consistent path for a given arc.

a) General graphs. We chose to vary two parameters: the number n of vertices of D and G (in the range 20, 30, 40, 50, 60), and the probability p that an edge between any given two vertices exists (in the range 0.05, 0.1, 0.15, 0.2). Taking any combination of these two parameters thus leads to 20 runs. We generated the undirected graphs G by using the Erdös-Rényi [9] random graphs generation method. We adapted this method, in order to construct random DAGs D, by randomly orienting the edges. For fixed n and p, we generated 100 couples (D, G). For each of these couples, we applied `AlgoH` and `AlgoBB` for 5 randomly chosen arcs, and computed the ratio ρ for each of the 5 corresponding instances. We then computed the number denoted $N_i^{(D,G,p,n)}$, $0 \leq N_i^{(D,G,p,n)} \leq 5$, of arcs whose $\rho \times 100$ belongs to I_i, with $1 \leq i \leq 10$ and $I_1 = [0, 10[, I_2 = [10, 20[, \ldots, I_{10} = [90, 100]$. We obtained a global result by computing, for each I_i, the value $m_i^{(n,p)} = \sum_{(D,G)} N_i^{(D,G,n,p)}$ i.e., the sum of $N_i^{(D,G,n,p)}$ for all 100 generated couples (D,G), with fixed n, p .

b) Scale-free graphs. We also applied our method on randomly generated scale-free graphs, since recent studies have shown that a large number of real-world networks tend to be scale-free (see e.g. [1]). In this experiment, we generated 100 couples (D,G) of 100 vertices, by using the public toolkit *NGCE* (http://ngce.sourceforge.net/). We observed that consistent paths are not abundant in scale-free graphs, thus we randomly chose, for each graph D, 10 arcs rather than 5 arcs as in the previous experiment. We then computed for each couple (D, G) and for each interval I_i, the number $N_i^{(D,G,n)}$, $0 \leq N_i^{(D,G,n)} \leq 10$, of arcs whose $\rho \times 100$ belongs to I_i and the global value $m_i^n = \sum_{(D,G)} N_i^{(D,G,n)}$ (here, n is fixed to 100).

We observe a very good behaviour of our heuristic AlgoH, since more than 90% of the input instances have a $\rho \times 100$ belonging to the interval $[90, 100]$ (see Figure 1). Also, it is very important to note the speed-up obtained by our heuristic with respect to the exact algorithm AlgoBB. For example, for the 500 instances of random graphs evaluated in the case $n = 60$ and $p = 0.2$ (Figure 1.**a**), Algorithm AlgoH was 11 times faster than AlgoBB.

5.2 Applying AlgoH on biological networks

We have also applied our heuristic AlgoH on real biological networks, in two different contexts, in order to verify that its results corroborate biological assumptions.

Metabolic pathway vs PPI network. A metabolic network is usually modeled by a directed graph (called a *reaction graph*) whose vertices are the reactions, and where there is an arc between two reactions if the first uses, as substrate, a product of the second. A metabolic network is represented in the *KEGG* database as a collection of functional modules (small networks) called *metabolic pathways*. Therefore, metabolic pathways can be modeled by DAGs [16]. A protein-protein interaction network (PPI) is modeled by an undirected graph whose vertices are the proteins, and there is an edge between each pair of physically interacting proteins. We applied our heuristic to extract automatically, in a metabolic pathway, a chain of reactions (i.e., a path) that are catalyzed by interacting proteins (i.e., a connected subgraph) in a PPI network. Such paths are biologically meaningful [7]: indeed, the authors of [7] divided the PPI network for the species *S. cerevisiae* into functional clusters and observed that proteins involved in successive reactions are generally more likely to interact than other protein pairs. They provided an example of a short path (of length 6) in the metabolic pathway *"Glycolysis/Gluconeogenesis"* corresponding to a functional cluster in the PPI network. In order to compare our results with theirs, we built the PPI graph G, of the same species *S. cerevisiae*, from the BioGRID database (http://thebiogrid.org/, version (v2.0.63)).We also constructed the metabolic pathway *"Glycolysis/Gluconeogenesis"* (graph D) from *KEGG* (*pathwaysce00010.xml*). We established the correspondence between the two graphs using the names of the genes which encode proteins that (a) catalyze reactions in the metabolic pathway and (b) interact in the PPI network. Notice that G does not have the same vertex set as D. In order to circumvent this difficulty, we used an additional graph G': the new graph G' is an undirected graph whose vertices are

the reactions ($V(G') = V(D)$) and there is an edge between two vertices $r_1, r_2 \in V(G')$, iff there are two interacting proteins $p_1, p_2 \in V(G)$ s.t. p_1 catalyzes r_1 and p_2 catalyzes r_2 (see also [2], where the construction of such a graph is detailed). Applying our heuristic ALGOH with the arc between vertices 1 and 23 as input, we automatically computed a (D, G')-consistent path of 12 vertices, inducing a connected subgraph of 20 proteins in the PPI network. Such a path includes those observed in [7].

Metabolic pathway vs Linear genome. We also applied ALGOH to automatically extract, in a metabolic pathway, a chain of reactions that are catalyzed by the products of adjacent genes in the genome. The genome network is modeled by an undirected graph whose vertices are the genes, and in which there is an edge between each adjacent pair of genes. The species under study was the bacterium *E. coli*. We built the linear sequence of genes (graph G) from the *NCBI* database. We constructed the metabolic pathway (graph D) from *KEGG* (*pathway eco00550.xml*). We established the correspondence between the two graphs using the names of the genes. Finally, as in the previous example, we constructed an additional undirected graph G' built on same vertex set of D. The longest (D, G')-consistent path we found by applying ALGOH for all the arcs of D is the same as the path found by Boyer et al. [3] (see Figure 3.b in [3]).

6 Conclusion

The SKEWGRAM problem belongs to a new type of subnetwork mining problems, arising from recent applications of biological, social or information networks: several graphs, of various types, represent different relations between objects, and a subset of objects is sought, with particular properties in each network. Due to an important set of parameters (the networks nature, the properties to fulfill, etc.), these problems are very complex. Still, they need good algorithmic solutions, since the size of the networks is often very large. In this paper, we studied the limits, in terms of graph classes (a graph representing a network), between difficult and easy cases, and we provided two algorithms, a reliable heuristic and an exact algorithm. We tested them on random data, in order to show the performances of our heuristics in terms of execution time and quality of the results. We also tested them on real data, in order to show their effectiveness on biological networks. Further studies should either investigate the complexity of SKEWGRAM in terms of approximability (on specific graph classes) and fixed parameter algorithms, or inexact variants of the

problem obtained, for instance by allowing small differences between the vertex set of the path in D and the connected set in G.

References

1. A.-L. Barabási. Scale-free networks: a decade and beyond. *Science*, 325:412–413, 2009.
2. G. Blin, G. Fertin, H. Mohamed-Babou, I. Rusu, F. Sikora, and S. Vialette. Algorithmic aspects of heterogeneous biological networks comparison. *In Proc. COCOA*, 6831:272–286, 2011.
3. F. Boyer, A. Morgat, L. Labarre, J. Pothier, and A. Viari. Syntons, metabolons and interactons: an exact graph-theoretical approach for exploring neighbourhood between genomic and functional data. *Bioinformatics*, 21(23):4209–4215, 2005.
4. H. Bunke. Graph matching: theoretical foundations, algorithms and applications. *In Proc. Vision Interface*, 2000:82–88, 2000.
5. D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Mining hidden community in heterogeneous social networks. *In Proc. of the 3rd International Workshop on Link Discovery*, pages 58–65, 2005.
6. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18:265–298, 2004.
7. P. Durek and D. Walther. The integrated analysis of metabolic and protein interaction networks reveals novel molecular organizing principles. *BMC Systems Biology*, 2(1), 2008.
8. S. Dzeroski and N. Lavrac. *Relational data mining*. Springer, 2001.
9. P. Erdös and A. Rényi. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.
10. A.-T. Gai, M. Habib, C. Paul, and M. Raffinot. Identifying common connected components of graphs. Technical Report RR-LIRMM-03016, LIRMM, 2003.
11. M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of* NP*-completeness*. W. H. Freeman, 1979.
12. B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. Pathblast: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 32:83–88, 2004.
13. Y. Matsuo, M. Hamasaki, H. Takeda, J. Mori, D. Bollegara, Y. Nakamura, T. Nishimura, K. Hasida, and M. Ishizuka. Spinning multiple social networks for semantic web. *In Proc. of the Twenty-First National Conference on Artificial Intelligence*, 2006.
14. R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. Mccuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *National Academy of Sciences*, 102(6):1974–1979, 2005.
15. R. Vicentini and M. Menossi. *Data mining and knowledge discovery in real life applications*. In-tech, Julio Ponce and Adem Karahoca edition, 2009.
16. S. Wernicke and F. Rasche. Simple and fast alignment of metabolic pathways by exploiting local diversity. *Bioinformatics*, 23:1978–1985, 2007.
17. E. Williams and D. J. Bowles. Coexpression of neighboring genes in the genome of *arabidopsis thaliana*. *Genome Research*, 14:1060–1067, 2004.