Bounded list injective homomorphism for comparative analysis of protein–protein interaction graphs

Isabelle Fagnot a,∗, Gaëlle Lelandais b, Stéphane Vialette c

a Laboratoire d’Informatique de l’Institut Gaspard Monge, UMR CNRS 8049, Université Paris-Est, 5 bd Descartes, 77454 Marne-la-Vallée Cedex 2, France

b Équipe de Bioinformatique Génomique et Moléculaire, INSERM U726, Université Paris 7, case 7113, 2 place Jussieu, 75251 Paris Cedex 05

c Laboratoire de Recherche en Informatique, UMR CNRS 8623, Université Paris-Sud, 91405 Orsay, France

Available online 9 June 2007

Abstract

Comparing genomic properties of multiple species at varying evolutionary distances is a powerful approach for studying biological and evolutionary principles. In the context of comparative analysis of protein–protein interaction graphs, we use a graph-based formalism to detect the preservation of a given protein complex. We show that the problem is polynomial-time solvable provided that each protein has at most two orthologs in the other species, but is hard for three. Also, we suggest ways to cope with hardness by proposing three translations of the problem into well-known combinatorial optimization problems, thereby allowing the use of many recent results in fast exponential-time algorithms. Motivated by the need for more accurate models, we conclude by giving and discussing three natural extensions of the problem.

© 2007 Published by Elsevier B.V.

Keywords: Protein interaction graph; Graph matching; List homomorphism

1. Introduction

High-throughput analysis make possible the study of protein–protein interactions at a genome-wise scale [11,14, 27]. The growing information on protein networks for different organisms lends to comparative analysis, which tries to determine the extent to which protein networks are conserved among species. Mounting evidence suggests that proteins that function together in a pathway or structural complex 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 [18].

Protein interactions identified on a genome-wide scale are commonly visualized as protein interaction graphs where proteins are vertices and interactions are edges [25]. Experimentally derived interaction networks can be extremely complex, so that it is a challenging problem to extract biological functions or pathways from them (even if some global
features have been found). However, biological systems are hierarchically organized into functional modules. Several methods have been proposed for identifying functional modules in protein–protein interaction graphs. As observed in [19], cluster analysis is an obvious choice of methodology for the extraction of functional modules from protein interaction networks. Tong et al. used \( k \)-cores\(^1 \) (highly connected complexes) to detect a novel nuclear network in yeast [26]. Another method is used in [23].

Comparative analysis of protein–protein interaction graphs aims at finding complexes that are common to different species. Kelley et al. [15] developed the program PathBlast, which aligns two protein–protein interaction graphs combining topology and sequence similarity. Sharan et al. [22] studied the complexes\(^2 \) that are conserved in \( S. \) Cerevisae and \( H. \) pylori, and found 11 significantly conserved complexes (several of these complexes match very well with prior experimental knowledge on complexes in yeast). They actually recasted the problem of searching for conserved complexes as a problem of searching for heavy subgraphs in an edge- and node-weighted graph, whose vertices are orthologous protein pairs.

We consider here a simpler combinatorial problem: given any complex that occurs in the protein–protein interaction graph of one species, find an occurrence (if any exists) of that complex in the protein–protein interaction graph of another species. Observe that we do not make any assumption about the topology of the complex, such as clique-like structure.

Graph matching is the most natural tool for studying such a problem. First, we should be tempted to recast the problem as the problem of searching for a subgraph isomorphism with respect to orthologous links (orthologous are strictly defined as genes that predate speciation and that code functionally equivalent proteins that arise from evolution. However, searching for an isomorphism is too strict). Indeed, no method\(^3 \) is able to identify all protein–protein interactions, so that we have to deal with false negatives. This suggests to search for a list injective homomorphism, i.e., an injective homomorphism with respect to orthologous links, of the complex (viewed as a graph) to the protein–protein interaction graph. The rationale of this is as follows. First, graph homomorphism only preserves adjacency, and hence can deal with false negatives. Second, injectivity is required in order to establish a bijective relationship between proteins in the complex and proteins in the occurrence. Finally, graph homomorphism with respect to orthologous links can be easily recasted as list homomorphism: a list of putative orthologs is associated to each protein (vertex) of the complex, and each such protein can only be mapped by the homomorphism to a protein occurring in its list.

We thus recast the problem as the problem of finding an injective graph homomorphism with respect to lists, i.e., orthologous links. In the context of comparative analysis of protein–protein interaction graphs, we need to impose drastic restrictions on the size of the lists. We will make the following important assumption: no protein has an unbounded number of orthologs in the other species, i.e., each list has a constant size and each protein has a constant number of occurrences among the lists. The present paper is devoted to analyzing the complexity of algorithms for solving that problem. Also, we investigate the counting related problem, i.e., how many occurrences (injective homomorphisms with respect to lists) of that complex are there in that protein–protein interaction graph?

This paper is organized as follows. Section 2 provides the general technical background for the discussion. In Section 3, we present a polynomial-time algorithm in case each vertex in \( G \) has at most two orthologs in \( H \). The problem is proven to be \( \text{NP} \)-complete in Section 4. Section 5 proposes three ways to cope with \( \text{NP} \)-hardness, and two extensions of the problem are discussed in Section 6. Section 7 concludes our work and suggests future directions.

2. Notations

A graph \( G \) consists of a finite set \( \mathcal{V}(G) = \{u_1, u_2, \ldots\} \) of elements called vertices together with a prescribed set \( \mathcal{E}(G) \) of undirected pairs of distinct vertices of \( \mathcal{V}(G) \). We abbreviate \( |\mathcal{V}(G)| \) to \( n(G) \), and \( |\mathcal{E}(G)| \) to \( m(G) \). The number \( n(G) \) is called the order of the graph. Every unordered pair \( e \in \mathcal{E}(G) \) of vertices \( u_i \) and \( u_j \) is called an edge of \( G \), written \( e = \{u_i, u_j\} \). We call \( u_i \) and \( u_j \) the endpoints of \( e \) and they are called adjacent vertices. The neighbor of a vertex \( u \in \mathcal{V}(G) \) is the set \( N(u) = \{v: \{u, v\} \in \mathcal{E}(G)\} \). An induced subgraph is a subset of the vertices of a graph

\(^1\) A \( k \)-core is a subgraph of the protein–protein interaction graph in which each protein is connected to at least \( k \) proteins of this subgraph.

\(^2\) They focused on dense, clique-like interaction patterns.

\(^3\) Although a number of methods are available for high-throughput analysis of protein–protein interactions, the most commonly used is the two-hybrid system.
Let \( V' \subseteq V(G) \). By \( G[V'] \) we denote the subgraph of \( G \) induced by \( V' \). A complete graph is a graph in which each pair of graph vertices is connected by a graph edge. We define a clique as any complete subgraph. An independent set of a graph \( G \) is a subset of \( V(G) \) such that no two vertices in the subset represent an edge of \( G \). The line graph \( L(G) \) of a graph \( G \) is a graph whose vertices are the edges of \( G \) and whose edges connect two vertices if and only if the corresponding edges of \( G \) meet at one or both endpoints. A bipartite graph is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. A graph is a split graph if its vertices can be partitioned into a clique and an independent set.

Let \( B \) be a bipartite graph with partition \( V(B) = V_1 \cup V_2, |V_1| \leq |V_2| \). A matching \( M \) on \( B \) is a subset of \( E(B) \) such that each vertex in \( V(B) \) is incident with no more than one edge in \( M \). A maximum matching is a matching whose cardinality is maximum among all matchings. A complete matching is a matching that saturates all of the vertices in \( V_1 \).

Let \( G \) and \( H \) be two graphs. A homomorphism of \( G \) to \( H \) is a mapping \( \theta : V(G) \rightarrow V(H) \) such that \( \{u, v\} \in E(G) \) implies \( \{\theta(u), \theta(v)\} \in E(H) \). Given lists \( L(u) \subseteq V(H), u \in V(G) \), a list homomorphism of \( G \) to \( H \) with respect to the lists \( L(u), u \in V(G) \), is a homomorphism \( \theta \) with the additional constraint that \( \theta(u) \in L(u) \) for all \( u \in V(G) \). For simplicity of notation, given lists \( L(u) \subseteq V(H), u \in V(G) \), we abbreviate \( \{u : v \in L(u)\} \rightarrow L^{-1}(v) \) if \( v \in V(H) \). Let \( G \) and \( H \) be two graphs. Lists \( L(u) \subseteq V(H), u \in V(G) \), are called \( (\mu_G, \mu_H) \)-bounded if the two following conditions hold true: (1) \( \max(|L(u)| : u \in V(G)) \leq \mu_G \) and (2) \( \max(|L^{-1}(v)| : v \in V(H)) \leq \mu_H \).

We are now in position to formally define the \((\mu_G, \mu_H)\)-MATCHING WITH ORTHOLOGIES problem that we will work with for the rest of the paper.

\((\mu_G, \mu_H)\)-MATCHING WITH ORTHOLOGIES

Input: Two graphs, \( G \) and \( H \), and \((\mu_G, \mu_H)\)-bounding lists \( L(u) \subseteq V(H), u \in V(G) \).

Question: Is there an injective homomorphism of \( G \) to \( H \) w.r.t. (with respect to) lists \( L(u), u \in V(G) \)?

Clearly, we may assume \( |L(u)| > 0, u \in V(G) \), and \( |L^{-1}(v)| > 0, v \in V(H) \). For now on, unless explicitly stated, we assume \( \mu_G = O(1) \) and \( \mu_H = O(1) \). We call \#(\(\mu_G, \mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES the related counting problem (we refer the reader to [17] for a complete treatment of the \#P class). Observe that for unbounded \( \mu_G \) and \( \mu_H \), the \((\mu_G, \mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES problem trivially contains the CLIQUE problem, and hence is \#P-complete [10].

3. Graph matching with at most two orthologs

We prove in this section that the \((\mu_G, \mu_H)\)-MATCHING WITH ORTHOLOGIES problem for \( \mu_G \leq 2 \) is polynomial-time solvable. The basic idea of the proof is to transform each input \((G, H, L)\) into a CNF formula \( \phi \) with at most two literals per clause in such a way that \( \phi \) is satisfiable if and only if there exists an injective homomorphism of \( G \) to \( H \) w.r.t. \( L(u), u \in V(G) \). Interestingly enough, in case \( \mu_G \leq 2 \), the problem remains polynomial-time solvable even if \( \mu_H \) is unbounded (see Proposition 3). Moreover, the bound on \( \mu_G \) is tight, since, as we shall see soon, the problem is \#P-complete for \( \mu_G \leq 3 \) and \( \mu_H = 1 \) (see Proposition 7).

Now we turn to describing our algorithm. Given an input \((G, H, L)\), the GRAPH-MATCHING-TO-2-CNF-SATISFIABILITY(G, H, L) algorithm (see Fig. 1) returns a 2-CNF formula which represents solutions of the \((\mu_G, \mu_H)\)-MATCHING WITH ORTHOLOGIES problem for \((G, H, L)\).

**Lemma 1.** Let \( \phi \) be the 2-CNF formula returned by Algorithm GRAPHMATCHING-TO-2-CNF-SATISFIABILITY(G, H, L). Then, \( \phi \) is satisfiable if and only if there exists an injective homomorphism of \( G \) to \( H \) w.r.t. \( L(u), u \in V(G) \).

**Proof.** Let \( \theta : V(G) \rightarrow V(H) \) be an injective homomorphism of \( G \) to \( H \) w.r.t. lists \( L(u), u \in V(G) \). We can check at once that the truth assignment \( f : X \rightarrow \{\text{true}, \text{false}\} \) defined by \( f(x_i) = \text{true} \) if and only if \( \sigma_i(\theta(i)) = x_i \) and \( f(x_i) = \text{false} \), is a satisfying assignment for \( \phi \).

Conversely, let \( f \) be any satisfying truth assignment for \( \phi \). Consider the mapping \( \theta : V(G) \rightarrow V(H) \) defined as follows: for all \( i \in V(G) \) with \( L(i) = \{j_1, j_2\} \), \( \sigma_i(j_1) = x_i \) and \( \sigma_i(j_2) = \overline{x_i} \),

\[
\theta(i) = \begin{cases} 
  j_1 & \text{if } f(x_i) = \text{true} \\
  j_2 & \text{if } f(x_i) = \text{false} 
\end{cases}
\]
GRAPH-MATCHING-TO-2-CNF-SATISFIABILITY \((G, H, \mathcal{L})\)

**Input:** Two graphs, \(G\) and \(H\), and \((\mu_G, \mu_H)\)-bounded lists \(\mathcal{L}(u) \subseteq \mathcal{V}(H), u \in \mathcal{V}(G)\). For the sake of clarity, we denote all vertices by the integers \(1, 2, \ldots, n(G)\).

**Output:** A 2-CNF formula \(\phi\) over a set of variables \(X\) defined by \(X = \{x_i \mid i \in \mathcal{V}(G)\}\).

1. (Each vertex of \(G\) has exactly one image in \(H\)) For all \(i \in \mathcal{V}(G)\) with \(\mathcal{L}(i) = \{j_1, j_2\}\), let \(\sigma_i(j_1) = x_i\) and \(\sigma_i(j_2) = \overline{x_i}\), arbitrary. In case \(\mathcal{L}(i) = \{j\}\), \(\sigma_i(j) = x_i\) and let \(\text{Choice}_G(i) = x_i\).

2. (Injectivity) For all \(j \in \mathcal{V}(H)\), with \(\mathcal{L}^{-1}(j) = \{i_1, i_2, \ldots, i_q\}\), let \(\text{Choice}_H(j)\) be the 2-CNF clause defined as follows:

\[
\text{Choice}_H(j) = \bigwedge_{1 \leq k < \ell \leq q} (\sigma_{i_k}(j) \lor \sigma_{i_{\ell}}(j))
\]

3. (Homomorphism, i.e., preservation of the edges of \(G\)) For all \(\{i, k\} \in \mathcal{E}(G)\) with \(\mathcal{L}(i) = \{j_1, j_2\}\), \(\sigma_i(j_1) = x_i\) and \(\sigma_i(j_2) = \overline{x_i}\), and \(\mathcal{L}(k) = \{\ell_1, \ell_2\}\), \(\sigma_k(\ell_1) = x_k\) and \(\sigma_k(\ell_2) = \overline{x_k}\), let \(\text{Edge}_H(i, k)\) be the 2-CNF formula corresponding to one of the following 16 cases (observe that the first case corresponds to the absence of solution while the last case corresponds to a trivial true clause):

4. Return the formula \(\phi\) obtained by conjunction of all the above defined formulae, i.e.,

\[
\phi = \bigwedge_{i \in \mathcal{V}(G), \mathcal{L}(i) = \{j\}} \text{Choice}_G(i) \land \bigwedge_{j \in \mathcal{V}(H)} \text{Choice}_H(j) \land \bigwedge_{\{i, k\} \in \mathcal{E}(G)} \text{Edge}_H(i, k).
\]

Fig. 1. Algorithm GRAPH-MATCHING-TO-2-CNF-SATISFIABILITY.

Thanks to \(\text{Choice}_H(j)\), \(j \in \mathcal{V}(H)\), \(\theta\) is an injective mapping. Furthermore, by \(\text{Edge}_H(i, k)\), \(\{i, k\} \in \mathcal{E}(G)\), \(\theta\) is a homomorphism of \(G\) to \(H\) w.r.t. lists \(\mathcal{L}(u), u \in \mathcal{V}(G)\).

**Lemma 2.** Given an input \((G, H, \mathcal{L})\), Algorithm GRAPH-MATCHING-TO-2-CNF-SATISFIABILITY \((G, H, \mathcal{L})\) runs in \(\mathcal{O}(n(G)^3 + m(G))\) time. This reduces to \(\mathcal{O}(n(G) + m(G))\) time in case \(\mu_H = \mathcal{O}(1)\).

**Proof.** Step 1 is a \(\mathcal{O}(n(G))\) time procedure. Step 2 is the most time-consuming part and runs in \(\mathcal{O}(n(H) \cdot n(G)^2)\) time which is \(\mathcal{O}(n(G)^3)\) since \(n(H) \leq 2n(G)\). Clearly, Step 3 can be done in \(\mathcal{O}(m(G))\) time. Summing up we obtain a \(\mathcal{O}(n(G)^3 + m(G))\) time algorithm. The last part of the lemma is proved by observing that Step 2 falls to \(\mathcal{O}(n(G))\) time in case \(\mu_H = \mathcal{O}(1)\).
Proposition 3. The \((\mu_G, \mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES problem for \(\mu_G \leq 2\) is solvable in \(\mathcal{O}(n(G)^3 + m(G))\) time. This reduces to \(\mathcal{O}(n(G) + m(G))\) time in case \(\mu_H = \mathcal{O}(1)\).

Proof. The algorithm is as follows:

1. Call Algorithm GRAPH-MATCHING-TO-2-CNF-SATISFIABILITY\((G, H, \mathcal{L})\), and let \(\phi\) be the returned 2-CNF formula.

2. Find a satisfying truth assignment \(f\) for \(\phi\).

3. Transform \(f\) into an injective homomorphism of \(G\) to \(H\).

According to Lemma 1, the algorithm is correct. What is left is to prove the time complexity. It is well known that the 2-SAT problem reduces to the graph problem of finding strongly connected components in the implication graph, and hence is solvable in linear time [1]. Step 3 is clearly a \(\mathcal{O}(n(G))\) time procedure. The time complexity thus follows from Lemma 2. \(\square\)

From a counting point of view, the \((\mu_G, \mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES problem for \(\mu_G \leq 2\) seems to be a much harder problem as shown in Proposition 4.

Proposition 4. The \((\mu_G, \mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES problem for \(\mu_G \leq 2\) is \#P-complete.

Proof. We exhibit a parsimonious reduction from the \#2-SAT problem, which is known to be \#P-complete [29], to the \((\mu_G, \mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES problem for \(\mu_G \leq 2\) and \(\mu_H = 1\). Let an arbitrary instance of the \#2-SAT problem be given by a 2-CNF formula \(\phi = c_1 \land c_2 \land \cdots \land c_m\) over variables \(X = \{x_1, x_2, \ldots, x_n\}\). Write \(c_k = \lambda_{k,1} \lor \lambda_{k,2}\).

Define two graphs \(G\) and \(H\) as follows.\(^4\) First, define a complete graph \(G\) with vertex set \(V(G) = \{c_{i,j} : 1 \leq i \leq m\) and \(1 \leq j \leq 2\). Second, define a graph \(H\) as follows:

\[
\begin{align*}
V(H) &= \{x_{i,j} : 1 \leq i \leq n\text{ and }1 \leq j \leq m\} \cup \{x_{i,j}^c : 1 \leq i \leq n\text{ and }1 \leq j \leq m\} \\
E(H) &= E_1(H) \cup E_2(H) \cup E_3(H) \cup E_4(H) \cup E_5(H) \cup E_6(H)
\end{align*}
\]

where

\[
E_1(H) = \{\{x_{i,j}, x_{i,k}\}, \{x_{i,j}^c, x_{i,k}^c\} : 1 \leq i \leq n\text{ and }1 \leq j < k \leq m\}
\]

\[
E_2(H) = \{\{x_{i,j}, x_{p,q}\} : 1 \leq i \leq n, 1 \leq p \leq n, i \neq p, 1 \leq j \leq m, 1 \leq q \leq m\text{ and }j \neq q\}
\]

\[
E_3(H) = \{\{x_{i,j}, x_{p,q}\} : 1 \leq i \leq n, 1 \leq p \leq n, i \neq p, 1 \leq j \leq m, 1 \leq q \leq m\text{ and }j \neq q\}
\]

\[
E_4(H) = \{\{x_{i,j}, x_{p,q}\} : 1 \leq i \leq n, 1 \leq p \leq n, i \neq p, 1 \leq j \leq m, 1 \leq q \leq m\text{ and }j \neq q\}
\]

\[
E_5(H) = \{\{x_{i,j}^c, x_{p,q}\} : 1 \leq i \leq n, 1 \leq p \leq n, i \neq p, 1 \leq j \leq m, 1 \leq q \leq m\text{ and }j \neq q\}
\]

We now turn to defining \(E_6(H)\). For each clause \(c_k = \lambda_{k,1} \lor \lambda_{k,2}\) of \(\phi\), we add three edges to \(E_6(H)\). These three edges depend on \(\lambda_{k,1}\) and \(\lambda_{k,2}\). The four cases (from left to right in the following figure) correspond to (1) \(\lambda_{k,1} = x_i\) and \(\lambda_{k,2} = x_j\), (2) \(\lambda_{k,1} = x_i^c\) and \(\lambda_{k,2} = x_j^c\), (3) \(\lambda_{k,1} = x_i\) and \(\lambda_{k,2} = x_j^c\), and (4) \(\lambda_{k,1} = x_i^c\) and \(\lambda_{k,2} = x_j\), respectively.

\[c_k = x_i \lor x_j\]
\[c_k = x_i \lor x_j^c\]
\[c_k = x_i^c \lor x_j\]
\[c_k = x_i^c \lor x_j^c\]

\(\text{To simplify notation, we introduce more vertices than needed, so that some vertices in } H \text{ never appear in any } \mathcal{L}(u), u \in V(G).\)
Our construction ends by defining (2, 1)-bounded lists \( L(u), u \in V(G) \). Corresponding to each clause \( c_k = \lambda_{k,1} \lor \lambda_{k,2} \) of \( \phi \), we define:
\[
L(c_k,1) = \{ x_i,k, \bar{x}_i,k \} \quad \text{provided that} \quad \lambda_{k,1} = x_i \text{ or } \lambda_{k,1} = \bar{x}_i
\]
\[
L(c_k,2) = \{ x_i,k, \bar{x}_i,k \} \quad \text{provided that} \quad \lambda_{k,2} = x_i \text{ or } \lambda_{k,2} = \bar{x}_i
\]

Suppose that there exists a satisfying truth assignment \( f \) for \( \phi \). Let \( \theta : V(G) \rightarrow V(H) \) be the injective mapping defined as follows:
\[
\forall c_k,j \in V(G), \quad \theta(c_k,j) = \begin{cases} x_i,k & \text{if } (\lambda_{k,j} = x_i \text{ or } \lambda_{k,j} = \bar{x}_i) \text{ and } f(x_i) = \text{true} \\ \bar{x}_i,k & \text{if } (\lambda_{k,j} = x_i \text{ or } \lambda_{k,j} = \bar{x}_i) \text{ and } f(x_i) = \text{false} \end{cases}
\]
Clearly, \( \theta(c_k,j) \in L(c_k,j), c_k,j \in V(G) \). It can be easily checked that \( \theta \) is an injective homomorphism of \( G \) to \( H \) w.r.t. lists \( L(u), u \in V(G) \).

Conversely, suppose that there exists an injective homomorphism \( \theta \) of \( G \) to \( H \) w.r.t. lists \( L(u), u \in V(G) \). First, we need the following fact.

Claim 5. For any variable \( x_i \in X \), and any two distinct vertices \( c_k,p \) and \( c_\ell,q \) of \( G \) such that \( \lambda_{k,p} \) and \( \lambda_{\ell,q} \) are positive or negative occurrences of a variable \( x_i \) (\( \lambda_{k,p} \) and \( \lambda_{\ell,q} \) could correspond to opposite literals), either (1) \( \theta(c_k,p) = x_i,k \) and \( \theta(c_\ell,q) = \bar{x}_i,k \) \( \phi \) is mapped by \( \theta \) to \( \bar{x}_i,k \) and (2) \( \theta(c_k,p) = \bar{x}_i,k \) and \( \theta(c_\ell,q) = x_i,k \) \( \phi \) is mapped by \( \theta \) to \( x_i,k \).

Proof. Suppose the claim is false, say \( \theta(c_k,p) = x_i,k \) and \( \theta(c_\ell,q) = \bar{x}_i,k \) (the other case leads to a similar contradiction). By construction, \( c_k,p \) and \( c_\ell,q \) are joined by an edge in \( G \) (since \( G \) is a complete graph), and \( x_i,k \) and \( \bar{x}_i,k \) are independent vertices in \( H \). Then it follows that \( \theta \) is not a homomorphism of \( G \) to \( H \), and this is the desired contradiction. \( \square \)

Thanks to the above claim, we can define a truth assignment for \( \phi \) as follows: for any variable \( x_i \in X \), write \( f(x_i) = \text{true} \) (resp. \( \text{false} \)) if each (positive or negative) occurrence \( \lambda_{k,p} \) of \( x_i \) in \( \phi \) is mapped by \( \theta \) to \( x_i,p \) (resp. \( \bar{x}_i,p \)).

We claim that \( f \) is a satisfying assignment for \( \phi \). Indeed, let \( c_k = \lambda_{k,1} \lor \lambda_{k,2} \) be any clause of \( \phi \). By construction, vertices \( c_k,1 \) and \( c_k,2 \) are joined by an edge in \( G \). Since \( \theta \) is a homomorphism of \( G \) to \( H \), it follows that the images of these two vertices by \( \theta \) have to be joined by an edge in \( H \). According to the definition of \( E_\theta(H) \), we can conclude that \( f \) satisfies clause \( c_k \). Therefore, \( f \) is a satisfying truth assignment for \( \phi \). \( \square \)

Proposition 6 gives the time-complexity of counting.

Proposition 6. The \( \#(\mu_G,\mu_H)\)-GRAPH MATCHING WITH ORTHOLOGIES problem for \( \mu_G \leq 2 \) is solvable in \( O(1.3247^n(G)) \) time.

Proof. The \#-SAT problem is solvable in \( O(1.3247^n) \) time where \( n \) is the number of variables [4]. The result thus follows from Lemmas 1 and 2. \( \square \)

We note that, according to a recent result of Williams [30], Proposition 6 reduces to \( \tilde{O}(1.2923^n(G)) \) expected time (\( \tilde{O} \) denotes average time-complexity).

4. Hardness result

We proved in the preceding section that the \( \phi_G,\phi_H \)-GRAPH MATCHING WITH ORTHOLOGIES problem for \( \mu_G \leq 2 \) and unbounded \( \mu_H \) is polynomial-time solvable. The bound on \( \mu_G \) is tight since, as we shall now see, the problem is already quite hard enough for \( \mu_G \leq 3 \) and \( \mu_H = 1 \).

Proposition 7. The \( \phi_G,\phi_H \)-GRAPH MATCHING WITH ORTHOLOGIES problem for \( \mu_G \leq 3 \) and \( \mu_H = 1 \) is \( \text{NP} \)-complete even if both \( G \) and \( H \) are bipartite graphs.
Proof. We shall transform the 3-SAT problem to the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem for $\mu_G \leq 3$ and $\mu_H = 1$ where both $G$ and $H$ are bipartite graphs. Let an arbitrary instance of the 3-SAT problem be given by a 3-CNF formula $\phi = c_1 \land c_2 \land \ldots \land c_m$ over variables $X = \{x_1, x_2, \ldots, x_n\}$. As usual, write $c_i = \lambda_{i,1} \lor \lambda_{i,2} \lor \lambda_{i,3}, 1 \leq i \leq m$.

Define two graphs $G$ and $H$ as follows: $\mathcal{V}(G) = V_1(G) \cup V_2(G) \cup V_3(G) \cup V_4(G)$ and $\mathcal{E}(G) = E_1(G) \cup E_2(G) \cup E_3(G)$ where

$\begin{align*}
V_1(G) &= \{u_i: 1 \leq i \leq n\} \\
V_2(G) &= \{s_i^G: 0 \leq i \leq n\} \\
V_3(G) &= \{c_i^G: 1 \leq i \leq m\} \\
V_4(G) &= \{t_i^G: 1 \leq i \leq m\} \\
E_1(G) &= \{(s_{i-1}^G, u_i): 1 \leq i \leq n\} \cup \{(s_i^G, u_i): 1 \leq i \leq m\} \\
E_2(G) &= \{(s_n^G, t_1^G): 1 \leq i \leq m\} \cup \{t_1^G, c_i^G): 1 \leq i \leq m\} \\
E_3(G) &= \{(u_i, c_j^G): 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}
\end{align*}$

and $\mathcal{V}(H) = V_1(H) \cup V_2(H) \cup V_3(H) \cup V_4(H)$ and $\mathcal{E}(H) = E_1(H) \cup E_2(H) \cup E_3(H)$ where

$\begin{align*}
V_1(H) &= \{x_i, \overline{x}_i: 1 \leq i \leq n\} \\
V_2(H) &= \{s_i^H: 0 \leq i \leq n\} \\
V_3(H) &= \{c_i^H: 1 \leq i \leq m \text{ and } 1 \leq j \leq 3\} \\
V_4(H) &= \{t_i^H: 1 \leq i \leq m\} \\
E_1(H) &= \{(s_n^H, x_i): 1 \leq i \leq n\} \cup \{(s_i^H, x_i): 1 \leq i \leq n\} \\
&\quad \cup \{(s_i^H, \overline{x}_i): 1 \leq i \leq n\} \cup \{(s_i^H, \overline{x}_i): 1 \leq i \leq n\} \\
E_2(H) &= \{(s_n^H, t_1^H): 1 \leq j \leq m\} \cup \{t_1^H, c_i^H): 1 \leq i \leq m \text{ and } 1 \leq j \leq 3\} \\
E_3(H) &= \{(x_i, c_{j+1}^H): 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq 3 \text{ and } \lambda_{j,k} \neq \overline{x}_i\} \\
&\quad \cup \{(\overline{x}_i, c_{j+1}^H): 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq 3 \text{ and } \lambda_{j,k} \neq \overline{x}_i\}
\end{align*}$

Clearly, both $G$ and $H$ are bipartite graphs. Our construction ends by defining $(3, 1)$-bounded lists $\mathcal{L}(u) \subseteq \mathcal{V}(H), u \in \mathcal{V}(G)$, as follows:

$\begin{align*}
\forall u_i \in V_1(G), & \quad \mathcal{L}(u_i) = \{x_i, \overline{x}_i\} \\
\forall s_i^G \in V_2(G), & \quad \mathcal{L}(s_i^G) = \{s_i^H\} \\
\forall c_i^G \in V_3(G), & \quad \mathcal{L}(c_i^G) = \{c_i^H: 1 \leq j \leq 3\} \\
\forall t_i^G \in V_4(G), & \quad \mathcal{L}(t_i^G) = \{t_i^H\}
\end{align*}$

It is easily seen that our construction can be carried on in polynomial-time.

Suppose that there exists a satisfying truth assignment $f$ for $\phi$. Again, observe that there is no loss of generality in assuming that each clause is satisfied by its first literal. Consider the injective mapping $\theta: \mathcal{V}(G) \to \mathcal{V}(H)$ defined as follows: (1) $\theta(u_i) = x_i$ (resp. $\overline{x}_i$) if $f(x_i) = \text{true}$ (resp. $\text{false}$), $u_i \in V_1(G)$, (2) $\theta(s_i^G) = s_i^H, s_i^G \in V_2(G)$, (3) $\theta(c_i^G) = c_i^H, c_i^G \in V_3(G)$, and (4) $\theta(t_i^G) = t_i^H, t_i^G \in V_4(G)$. It can be easily verified that $\theta$ is an injective homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in \mathcal{V}(G)$.

Conversely, suppose that there exists an injective homomorphism $\theta$ of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in \mathcal{V}(G)$. Define a truth assignment $f$ for $\phi$ as follows: $\phi(x_i) = \text{true}$ (resp. $\text{false}$) if $\theta(u_i) = x_i$ (resp. $\overline{x}_i$), $x_i \in X$. We claim that $f$ is a satisfying truth assignment for $\phi$. Indeed, we first observe that $\{u_i: 1 \leq i \leq n\} \cup \{c_i^G: 1 \leq i \leq m\}$ induces a complete bipartite graph in $G$. Since $\theta$ is a homomorphism of $G$ to $H$, then it follows that $\{\theta(u_i): 1 \leq i \leq n\} \cup \{\theta(c_i^G): 1 \leq i \leq m\}$ induces a complete bipartite graph in $H$. Therefore, thanks to $E_3(H)$, we can conclude that $f$ is a satisfying truth assignment for $\phi$. □
It is worth noticing that Proposition 7 holds true even if both $G$ and $H$ are split graphs (see [8] for details).

5. Coping with NP-hardness

5.1. Introduction

We proved that the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is polynomial-time solvable for $\mu_G \leq 2$, but is NP-complete for $\mu_G > 2$. Beside this latter result, it is worthwhile keeping in mind that intractability must be coped with and problems must be solved in practical applications. We present here an approach based on translating the problem into another combinatorial problem. Indeed, decades of research have led to comprehensive advances in the understanding of many combinatorial optimization problems from which we can greatly benefit. In the present paper, the emphasis is clearly on “efficient” exponential-time algorithms for three well-known combinatorial optimization problems.

Before giving the translations, we want to insist on the fact that our main goal is not concerned with comparing the three translations (in particular, we don’t draw any practical conclusion from the time-complexity we obtained), but only to propose practical ways of coping with NP-hardness.

5.2. Maximal matching

We begin with a simple enumeration algorithm. Let $(G, H, \mathcal{L})$ be an instance of the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem. Construct a bipartite graph $B_{G,H,\mathcal{L}} = B$ as follows: $\mathcal{V}(B) = \mathcal{V}(G) \cup \mathcal{V}(H)$ and $\mathcal{E}(B) = \{[u, v]: u \in \mathcal{V}(G), v \in \mathcal{V}(H) \text{ and } v \in \mathcal{L}(u)\}$. Loosely speaking, $B$ is a graph-encoding of the lists $\mathcal{L}$; observe however that $B$ is a bipartite graph of bounded degree since $\mu_G = \mathcal{O}(1)$ and $\mu_H = \mathcal{O}(1)$. Most of the interest in $B$ stems from the following easy remark.

**Remark 5.1.** Let $\theta : \mathcal{V}(G) \rightarrow \mathcal{V}(H)$ be an injective homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in \mathcal{V}(G)$. Then, $\mathcal{M} = \{[u, \theta(u))]: u \in \mathcal{V}(G)\} \subseteq \mathcal{E}(B)$ is a complete matching in $G$.

**Proposition 8.** The $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is solvable in $\mathcal{O}(n(G) \sqrt{n(G)}) + M(m(G) + n(G))$ time and $\mathcal{O}(n(G))$ space, where $M$ is the number of maximum matchings in $B = B_{G,H,\mathcal{L}}$.

**Proof.** The algorithm is as follows:

Correctness is immediate (we are only concerned with maximum matchings $\mathcal{M}$ of size $n(G)$, i.e., complete matchings). We now turn to proving the time-complexity. T. Uno gave an algorithm for enumerating all maximum matching in a bipartite graph in $\mathcal{O}(m(B) \sqrt{n(B)} + n(B) \cdot M)$ time where $M$ is the number of maximum matchings in $B$ [28]. Furthermore, each iteration of the algorithm is $\mathcal{O}(m(B))$ time. The proposition thus follows from $n(B) = n(G) + n(H) \leq n(G)(\mu_G + 1) = \mathcal{O}(n(G))$ and $m(B) \leq \mu_G \cdot n(G) = \mathcal{O}(n(G))$. □

Of course, $M$ can be as large as $\min\{n(G), n(H)\}$, and hence Proposition 8 is actually an exponential-time algorithm. Despite its bad time-complexity, Proposition 8 will prove to be useful in Section 6.1. Moreover, the algorithm of Proposition 8 can be easily modified so that it effectively checks each complete matching in $B$, and hence each injective mapping $\theta : \mathcal{V}(G) \rightarrow \mathcal{V}(H)$. We have thus proved the following: the #$(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is solvable within the same time complexity as in Proposition 8 (we observe that this algorithm is actually an enumeration algorithm).

5.3. Independent set

We use here a different idea. Indeed, instead of enumerating all possible solutions (by means of complete matchings in the bipartite graph $B$) to possibly find a solution, we construct a more elaborate graph $G'$ in such a way that any sufficiently large independent set in $G'$ can be transformed into a solution of our main problem.
We now turn to the construction of $G'$. Let $(G, H, \mathcal{L})$ be an instance of the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem. Construct a graph $G'$ as follows: $\mathcal{V}(G') = \{(u, v) \in \mathcal{V}(G) \times \mathcal{V}(H): v \in \mathcal{L}(u)\}$ and $\mathcal{E}(G') = E_1 \cup E_2$ where

\[
E_1 = \{(u, v), (u', v')\}: u = u' \text{ or } v = v' \}
\]
\[
E_2 = \{(u, v), (u', v')\}: \{u, u'\} \in \mathcal{E}(G) \text{ and } \{v, v'\} \notin \mathcal{E}(H) \}
\]

Clearly, $G'$ has at most $\mu_G \cdot n(G)$ vertices and is a super-graph of the line graph of $B_{G, H, \mathcal{L}}$. Connection with the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is given by the following lemma.

**Lemma 9.** Let $G'$ be the graph described above. Then, there exists an independent set of size $n(G)$ in $G'$ if and only if there exists an injective homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in \mathcal{V}(G)$.

**Proof.** Suppose there exists an injective homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in \mathcal{V}(G)$. Let $V' \subseteq \mathcal{V}(G')$ be the subset defined as follows: for each $u \in \mathcal{V}(G)$, add vertex $(u, \theta(u))$ to $V'$. Clearly, $|V'| = n(G)$. Since $\theta$ is injective, no two vertices of $V'$ are connected by an edge in $E_1$. Furthermore, $\theta$ is edge preserving, and hence no two vertices of $V'$ are connected by an edge in $E_2$. Then it follows that $V'$ is an independent set of size $n(G)$ in $G'$.

Conversely, suppose there exists an independent subset $V' \subseteq \mathcal{V}(G')$ of size $n(G)$ in $G'$. Define a mapping $\theta: \mathcal{V}(G) \to \mathcal{V}(H)$ as follows: for each $(u, v) \in V'$, we take $\theta(u) = v$. Thanks to $E_1$, $\theta$ is an injective mapping w.r.t. lists $\mathcal{L}(u), u \in \mathcal{V}(G)$. Furthermore, according to $E_2$, $\theta$ is a homomorphism of $G$ to $H$, i.e., edge preserving mapping. □

Tarjan and Trojanowski [24] were the first to break through the 2$^n$ barrier for the MAXIMUM INDEPENDENT SET problem. This time-complexity has been improved many times [2,20,21]. We use the more recent results in the following proposition.

**Proposition 10.** The $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is solvable in $O(1.1889^m(G))$ time and exponential-space, or, in $O(1.2025^m(G))$ time and polynomial-space.

**Proof.** The algorithm is as follows: Construct the graph $G'$ as detailed above, and find a maximum independent set in it. Correctness follows directly from Lemma 9. The MAXIMUM INDEPENDENT SET problem is solvable in $O(1.1889^m(G))$ time and exponential-space, or in $O(1.2025^m(G))$ time and polynomial-space [21]. Neglecting the time needed for the construction of $G'$, the proposition is proved. □

### 5.4. Satisfiability

We present in this subsection an algorithm for solving the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem which is based on Boolean formula satisfiability. The algorithm is, however, completely different from the one in Section 3. Indeed, since we allow more than two orthologs in $H$ for each vertex in $G$, we cannot associate a binary choice to each list $\mathcal{L}(u), u \in \mathcal{V}(G)$. The main idea here is to associate to each pair $(u, v) \in \mathcal{V}(G) \times \mathcal{V}(H)$ with $v \in \mathcal{L}(u)$, a boolean variable which is true if and only if we have $\theta(u) = v$ in the injective homomorphism $\theta$ of $G$ to $H$ we are looking for. A detailed description of Algorithm Graph-Matching-To-Satisfiability is given in Fig. 2.

**Lemma 11.** Algorithm Graph-Matching-To-Satisfiability$(G, H, \mathcal{L})$ runs in $O(n(G) + m(G) + n(H))$ time.

**Proof.** Clearly, Step 1 (resp. Step 2) is a $O(n(G))$ (resp. $O(n(H))$) time procedure. We now turn to Step 3. Observe that for each $[i, k] \in \mathcal{E}(G)$, $\text{Edge}_{\mu}(i, k)$ is a DNF formula which contains at most $\mu_G^2 = O(1)$ clauses, and hence can be transformed into a CNF formula in $O(1)$ time by standard techniques. Therefore, Step 3 is a $O(m(G))$ time procedure, and the lemma follows. □

**Lemma 12.** Let $\phi$ be the CNF formula returned by Algorithm Graph-Matching-To-Satisfiability$(G, H, \mathcal{L})$. Then, $\phi$ contains $O(n(G))$ variables and $O(n(G) + m(G))$ clauses.
Graph-Matching-To-Satisfiability($G, H, \mathcal{L}$)

**Input:** Two graphs, $G$ and $H$, and ($\mu_G$, $\mu_H$)-bounded lists $\mathcal{L}(u) \subseteq V(H)$, $u \in V(G)$ (For the sake of clarity, we denote all vertices by the integers $1, 2, \ldots, n(G)$).

**Output:** A Boolean formula $\phi$ over a set of variables $X$ defined by $X = \{x_{i,j} : 1 \leq i \leq n(G) \text{ and } j \in \mathcal{L}(i)\}$.

(1) *(Each vertex of $G$ has exactly one image in $H$)* For all $i \in V(G)$ with $\mathcal{L}(i) = \{j_1, j_2, \ldots, j_p\}$, let $\text{Choice}_G(i)$ be the $p$-CNF formula defined as follows

$$
\text{Choice}_G(i) = (x_{i,j_1} \lor x_{i,j_2} \lor \cdots \lor x_{i,j_p}) \land \bigwedge_{1 \leq k < \ell \leq p} (\overline{x_{i,j_k}} \lor \overline{x_{i,j_\ell}}).
$$

(2) *(Injectivity)* For all $j \in V(H)$ with $\mathcal{L}^{-1}(j) = \{i_1, i_2, \ldots, i_q\}$, let $\text{Choice}_H(j)$ be the 2-CNF formula defined as follows

$$
\text{Choice}_H(j) = \bigwedge_{1 \leq k < \ell \leq q} (\overline{x_{i_k,j}} \lor \overline{x_{i_\ell,j}}).
$$

(3) *(Homomorphism, i.e., preservation of the edges of $G$)* For all $[i, k] \in E(G)$ let $\text{Edge}_G(i, k)$ be the DNF formula defined as follows

$$
\text{Edge}_G(i, k) = \bigvee_{j \in \mathcal{L}(i), k \in \mathcal{L}(k) \{j\} \in E(H)} (x_{i,j} \land x_{k,j}).
$$

Transform each DNF formula $\text{Edge}_G(i, k)$ into an equivalent CNF formula.

(4) Return the formula $\phi$ obtained by conjunction of all the above defined formulae, i.e.,

$$
\phi = \bigwedge_{i \in V(G)} \text{Choice}_G(i) \land \bigwedge_{j \in V(H)} \text{Choice}_H(j) \land \bigwedge_{[i, k] \in E(G)} \text{Edge}_G(i, k).
$$

Fig. 2. Algorithm Graph-Matching-To-Satisfiability.

**Proof.** Clearly, $|X| \leq \mu_G \cdot n(G)$, and hence $\phi$ contains $O(n(G))$ variables. For each $i \in V(G)$, $\text{Choice}_G(i)$ is a CNF formula with at most $\frac{1}{2}\mu_G (\mu_G - 1) + 1$ clauses. For each $j \in V(H)$, $\text{Choice}_H(j)$ is a CNF formula with at most $\frac{1}{2}\mu_H (\mu_H - 1)$ clauses. For each $[i, k] \in E(G)$, $\text{Edge}_G(i, k)$ is a DNF formula with at most $\mu_G^2$ clauses. By standard technique, it follows that $\text{Edge}_G(i, k)$ can be transformed into a CNF formula with at most $2^\mu_G$ clauses. Summing-up, $\phi$ contains at most

$$
n(G) \left( \frac{\mu_G (\mu_G - 1)}{2} + 1 \right) + n(H) \cdot \frac{\mu_H (\mu_H - 1)}{2} + m(G) \cdot 2^\mu_G
$$

$$
\leq n(G) \left( \frac{\mu_G (\mu_G - 1)}{2} + \frac{\mu_G \mu_H (\mu_H - 1)}{2} + 1 \right) + m(G) \cdot 2^\mu_G,
$$

which is $O(n(G) + m(G))$. □

**Lemma 13.** Let $\phi$ be the CNF formula returned by Algorithm Graph-Matching-To-Satisfiability($G, H, \mathcal{L}$). Then, $\phi$ is satisfiable if and only if there exists an injective homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in V(G)$.

**Proof.** Let $\theta : V(G) \rightarrow V(H)$ be an injective homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in V(G)$. We can check at once that the truth assignment $f : X \rightarrow \{\text{true}, \text{false}\}$ defined by $f(x_{i,j}) = \text{true}$ if and only if $\theta(i) = j, i \in V(G)$, is a satisfying assignment for $\phi$.

Conversely, let $f$ be any satisfying truth assignment for $\phi$. First, thanks to $\text{Choice}_G(i), i \in V(G)$, exactly one variable of $\{x_{i,j} : j \in \mathcal{L}(i)\}$ is set to true by $f$. We thus can define a mapping $\theta : V(G) \rightarrow V(H)$ as follows: for all $i \in V(G)$, $\theta(i) = j$ if and only if $f(x_{i,j}) = \text{true}$. According to $\text{Choice}_H(j), j \in V(H)$, $\theta$ is an injective mapping. Furthermore, by $\text{Edge}_G(i, k), [i, k] \in E(G)$, $\theta$ is a homomorphism of $G$ to $H$ w.r.t. lists $\mathcal{L}(u), u \in V(G)$. □
According to the above lemma, we are thus reduced to solving $\phi$. The SAT problem has seen much theoretical interest as the canonical NP-complete problem [3]. This research has resulted in the development of several satisfiability algorithms that have seen practical success (see [12] for a nice review). For a given instance, (complete) satisfiability solvers can either find a solution or prove that no solution exists. Of course, we can solve an instance of the SAT problem by simply testing each $2^n$ assignment, where $n$ is the number of variables. We can do better if we insist on the number of clauses as the input parameter. The original (non-brute force) algorithm for solving the SAT problem is often attributed to Davis and Putnam [6]. This algorithm has been improved many times. One of the best known bounds using the Davis–Putnam procedure (and using the number of clauses as parameter) was obtained by Hirsch [13] and is used in the following proposition.

**Proposition 14.** The $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is solvable in $O(1.2388^{n(G)} + m(G))$ time.

**Proof.** Hirsch proposed a $O(1.2288^m)$ time algorithm for the SAT problem, where $m$ is the number of clauses [13]. The proposition thus follows from Lemmas 12 and 13.

Surprisingly, if we insist on the number of variables as the input parameter for solving the SAT problem, no better algorithm than the brute-force $O(2^n)$ time one is known. However, using deterministic local search combined with covering codes, significant improvement has been recently proposed [5]. We use that latter result in the following proposition.

**Proposition 15.** The $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem is solvable in time $(2 - 2/(\mu_G + 1))^{n(G)}$ up to a polynomial factor.

**Proof.** Dantsin et al. proposed an algorithm running in time $(2 - 2/(k + 1))^n$ up to a polynomial factor for the SAT problem, where $n$ is the number of variables [5]. The proposition thus follows from Lemmas 12 and 13, and from the fact that each clause in $\phi$ has size at most $\mu_G$. □

Call a CNF formula monotone, if each clause contains either only positive literals or only negative literals. Observe that the CNF formula $\phi$ returned by algorithm Graph-Matching-To-Satisfiability($G$, $H$, $\mathcal{L}$) is monotone and it could be possible that algorithms with better exponential worst-case running time exist for solving the MONOTONE $k$-SAT problem. However, we are not aware of such an algorithm.

From a practical point of view, boolean satisfiability solvers which are available today are the result of decades of research and are deemed to be among the faster NP-complete problem specific solvers. The latest generation of SAT solver generally have three key features: randomization of variable selection, backtracking search and some form of clause learning. Beside the theoretical results we achieved, we want to insist on the fact that Algorithm Graph-Matching-To-Satisfiability($G$, $H$, $\mathcal{L}$) can be used as a pre-processing step before using fast heuristic algorithms for solving the SAT problem such as Zchaff (an efficient implementation of the Chaff algorithm [16]).

6. Two extensions

Aiming at more accurate models, we propose in this section two natural extensions of the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem. We prove the first to be fixed-parameter tractable with respect to the number of vertices in $G$ having at least two orthologs in $H$, and the second to be W[1]-hard.

6.1. Bounding the number of ambiguous vertices

In practical applications, many vertices in $G$ may have only one ortholog in $H$. This suggests the following question: is the $(\mu_G, \mu_H)$-GRAPH MATCHING WITH ORTHOLOGIES problem parametrically tractable with respect to the number of vertices in $G$ having at least two orthologs in $H$? The answer is positive and is detailed in the following proposition.
Proposition 16. The \((\mu_G, \mu_H)\)-Graph Matching With Orthologies problem parameterized by \(k = |\{u \in \mathcal{V}(G) : |\mathcal{L}(u)| > 1\}|\) is solvable in \(O(k\mu_G^2(n(G) + m(G)))\) time, and hence is fixed-parameter tractable.

Proof. Let \(V'_G = \{u \in \mathcal{V}(G) : |\mathcal{L}(u)| > 1\}\) and \(V'_H = \{v \in \mathcal{V}(H) : \exists u \in V'_G \text{ s.t. } v \in \mathcal{L}(u)\}\). Let \((G', H', \mathcal{L}')\) be the input instance of the \((\mu_G, \mu_H)\)-Graph Matching With Orthologies problem induced by \(V'_G\) and \(V'_H\), i.e., \(G' = G[V'_G], H' = H[V'_H]\) and \(\mathcal{L}'\) is the restriction of \(\mathcal{L}\) to \(V'_G\). The algorithm is as follows:

The correctness of the algorithm is obvious. We now turn to proving the time-complexity. Let \(B = B_{G', H', \mathcal{L}'}\) be the bipartite graph defined as in Section 5.2. Clearly, injective mappings \(\theta' : \mathcal{V}(G') \rightarrow \mathcal{V}(H')\) are in bijection with complete matchings in \(B\). According to [28] (see also Proposition 8), maximum matchings in \(B\) can be enumerated in \(O(|B| \cdot \sqrt{n(G')} + n(B) \cdot M)\) time where \(M\) is the number of maximum matchings in \(B\). Combining this with \(n(B) \leq k(\mu_G + 1), m(B) \leq \mu_G k\) and \(M \leq \mu_G^k\), we obtain a \(O(k\mu_G^k)\) time procedure for enumerating all injective mappings \(\theta' : \mathcal{V}(G') \rightarrow \mathcal{V}(H')\). The construction of \(\theta\) is a \(O(n(G))\) time procedure. Next, checking whether \(\theta\) is an injective homomorphism of \(G\) to \(H\) w.r.t. lists \(\mathcal{L}\) can be done in \(O(m(G))\) time. Summing up, we obtain a \(O(k\mu_G^k(n(G) + m(G)))\) time algorithm. □

6.2. Weighted orthology links

This subsection deals with weighted orthology links. Indeed, from a more realistic point of view, BLAST scores (and synteny) are used to infer orthologous relationships, so that we may assume that orthology links are ranked according to their confidence. The advantages for such an extension are two-fold. First, the more complete the model is, the more accurate it becomes. Second, such an addition turns the \((\mu_G, \mu_H)\)-Graph Matching With Orthologies problem into an optimization one, thereby allowing us to start investigating approximation and parameterized properties of the problem.

For simplicity of notation, weighted orthology links are given in the form of a weight function \(\omega : \mathcal{V}(G) \times \mathcal{V}(H) \rightarrow \mathbb{N}^+\) such that \(\omega(u, v) > 0\) only if \(v \in \mathcal{L}(u)\). Observe that this requirement allows \(\omega(u, v) = 0\) even if \(v \in \mathcal{L}(u)\). Formally, the problem is thus defined as follows.

**Weighted \((\mu_G, \mu_H)\)-Matching With Orthologies**

**Input:** Two graphs \(G\) and \(H\), \((\mu_G, \mu_H)\)-bounding lists \(\mathcal{L}(u) \subseteq \mathcal{V}(H), u \in \mathcal{V}(G)\), a weight function \(\omega : \mathcal{V}(G) \times \mathcal{V}(H) \rightarrow \mathbb{N}^+\) such that \(\omega(u, v) > 0\) only if \(v \in \mathcal{L}(u)\), and a positive integer \(k\).

**Question:** Is there an injective homomorphism \(\theta\) of \(G\) to \(H\) w.r.t. lists \(\mathcal{L}(u), u \in \mathcal{V}(G)\), such that \(\sum_{u \in \mathcal{V}(G)} \omega(u, \theta(u)) \geq k\)?

**Parameter:** \(k\)

In contrast to Section 3, we prove that the **Weighted \((\mu_G, \mu_H)\)-Matching With Orthologies** problem is hard for \(\mu_G \leq 2\) and \(\mu_H = 1\). (The result is given in the form of fixed-parameter intractability.)

Proposition 17. The **Weighted \((\mu_G, \mu_H)\)-Graph Matching With Orthologies** problem for \(\mu_G \leq 2\) and \(\mu_H = 1\) is **W[1]**-hard even if all weights are either 0 or 1.

Proof. We shall transform the **Independent Set** problem, which is known to be **W[1]**-complete [7], to the **Weighted \((\mu_G, \mu_H)\)-Graph Matching With Orthologies** problem with \(\mu_G = 2\) and \(\mu_H = 1\). Let \((G, k)\) be an instance of the **Independent Set** problem. With the notations \(n = n(G)\) and \(\mathcal{V}(G) = \{u_i : 1 \leq i \leq n\}\), define a graph \(H\) as follows:

\[
\mathcal{V}(H) = \{v_i : 1 \leq i \leq n\} \cup \{v'_i : 1 \leq i \leq n\}
\]

\[
\mathcal{E}(H) = \bigcup_{1 \leq i \leq j \leq n} \{\{v_i, v'_j\}, \{v'_j, v'_i\}, \{v'_i, v_j\} : \{u_i, u_j\} \in \mathcal{E}(G)\}
\]

Construct lists \(\mathcal{L}(u) \subseteq \mathcal{V}(H), u \in \mathcal{V}(G)\), defined as follows: \(\mathcal{L}(u_i) = \{v_i, v'_i\}, u_i \in \mathcal{V}(G)\), and a weight function \(\omega : \mathcal{V}(G) \times \mathcal{V}(H) \rightarrow \mathbb{N}\) defined by:

\[
\omega(u, v) = \begin{cases} 1 & \text{if } u = u_i \text{ and } v = v_i \text{ for some } 1 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}
\]
Clearly, \( \omega(u,v) > 0 \) only if \( v \in L(u) \), and hence \( \omega \) satisfies the requirement of the problem. We claim that there exists an independent set of size \( k \) in \( G \) if and only if there exists an injective homomorphism \( \theta \) of \( G \) to \( H \) w.r.t. lists \( L(u) \), \( u \in V(G) \), such that \( \sum_{u \in V(G)} \omega(u, \theta(u)) \geq k \).

Suppose that there exists an independent set \( V' \subseteq V(G) \) of size \( k \) in \( G \). Let \( \theta : V(G) \to V(H) \) be the injective mapping defined by \( \theta(u_i) = v_i \) if \( u_i \in V' \) and \( \theta(u_i) = v_j \) if \( u_i \notin V' \), \( u_j \in V(G) \). One can easily see that \( \theta \) is an injective homomorphism of \( G \) to \( H \) w.r.t. lists \( L(u) \), \( u \in V(G) \), such that \( \sum_{u \in V(G)} \omega(u, \theta(u)) = k \).

Conversely, suppose that there exists an injective homomorphism \( \theta \) of \( G \) to \( H \) w.r.t. lists \( L(u) \), \( u \in V(G) \), such that \( \sum_{u \in V(G)} \omega(u, \theta(u)) \geq k \). Let \( V' \subseteq V(G) \) be the subset of vertices of \( G \) defined by \( u \) in \( V' \) if and only if \( \omega(u, \theta(u)) = 1 \), \( u \in V(G) \). Clearly, \( |V'| \geq k \). Let \( u_i \) and \( u_j \) be two distinct vertices in \( V' \). According to the definition of \( \omega \), we must have \( \theta(u_i) = v_i \) and \( \theta(u_j) = v_j \). Observe that \( \{v_i, v_j\} \notin E(H) \), and hence, since \( \theta \) is a homomorphism of \( G \) to \( H \), we must have \( \{u_i, u_j\} \notin E(G) \). Therefore, \( V' \) is an independent set of size at least \( k \) in \( G \).

It remains open, however, to precisely determine how approximable is the \( \text{WEIGHTED } (\mu_G, \mu_H) \)-\textsc{Graph Matching With Orthologies} problem.

### 7. Conclusion

In the context of comparative analysis of protein–protein interaction graphs, we investigated the complexity of finding an injective homomorphism of a graph \( G \) to a graph \( H \) with respect to bounded lists. We proved that the problem is polynomial-time solvable if each vertex of \( G \) has at most two orthologs in \( H \), and is \( \text{NP} \)-complete for three. Also, we gave three translations of that problem into well-known combinatorial optimization problems. We mention that the approximate version of the \( (\mu_G, \mu_H) \)-\textsc{Graph Matching With Orthologies} problem is considered in [9].

### References


