On The Subgraph Epimorphism Problem

Steven Gay[a], François Fages[a,*], Thierry Martinez[a], Sylvain Soliman[a], Christine Solnon[b,c]

[a] *EPI Contraintes, Inria Paris-Rocquencourt, Domaine de Voluceau 78150 Rocquencourt, France*
[b] *Université de Lyon, CNRS*
[c] *INSA-Lyon, LIRIS, UMR5205, F-69621, France*

Abstract

In this paper we study the problem of deciding the existence of a subgraph epimorphism between two graphs. Our interest in this variant of graph matching problem stems from the study of model reductions in systems biology, where large systems of biochemical reactions can be naturally represented by bipartite digraphs of species and reactions. In this setting, model reduction can be formalized as the existence of a sequence of vertex deletion and merge operations that transforms a first reaction graph into a second graph. This problem is in turn equivalent to the existence of a subgraph (corresponding to delete operations) epimorphism (i.e. surjective homomorphism, corresponding to merge operations) from the first graph to the second. In this paper, we study theoretical properties of subgraph epimorphisms in general directed graphs. We first characterize subgraph epimorphisms (SEPI), subgraph isomorphisms (SISO) and graph epimorphisms (EPI) in terms of graph transformation operations. Then we study the graph distance measures induced by these transformations. We show that they define metrics on graphs and compare them. On the algorithmic side, we show that the SEPI existence problem is NP-complete by reduction of SAT, and present a constraint satisfaction algorithm that has been successfully used to solve practical SEPI problems on a large benchmark of reaction graphs from systems biology.

Keywords: Subgraph epimorphim, Model reduction, Graph distance, Constraint solving, Systems biology.

1. Introduction

Our interest in subgraph epimorphisms stems from the study of model reductions in systems biology, where large systems of biochemical reactions can be naturally represented by bipartite digraphs of species and reactions [14, 10]. In this setting, one can define a very general notion of model reduction as a particular form of graph transformation and use it to compare models in systems biology model repositories [8].

Let us consider, for example, the reduction of Michaelis-Menten in figure 1. The left-hand side graph is a detailed model composed of three reactions where an enzyme E binds in a reversible manner to a substrate S to form a complex ES and release a product P. The right-hand side graph reduces this system to a single reaction catalyzed by the enzyme.

The reduced graph can be obtained from the source graph by a sequence of delete and merge operations on species and reaction vertices. These transformations can typically be justified in

*Corresponding author. Phone: +33 1 39 63 57 09. Fax: +33 1 39 63 54 69.
Email addresses: Steven.Gay@inria.fr (Steven Gay), Francois.Fages@inria.fr (François Fages), Thierry.Martinez@inria.fr (Thierry Martinez), Sylvain.Soliman@inria.fr (Sylvain Soliman), Christine.Solnon@liris.cnrs.fr (Christine Solnon)

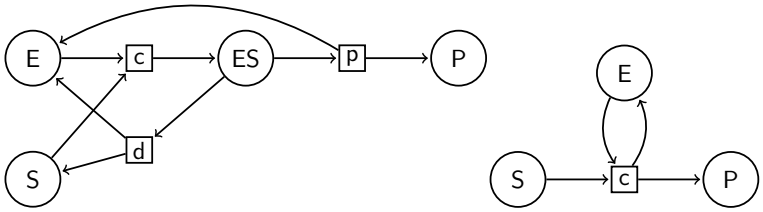


Figure 1: A catalytic mechanism, and the Michaelis-Menten reduced mechanism

chemistry by considering: (i) reaction deletions for slow reverse reactions, (ii) reaction mergings for reaction chains with a limiting reaction, (iii) molecular species deletions for species in excess and (iv) molecular mergings for quasi-steady state approximations.

This operational view of graph reduction by graph transformation operations is equivalent to the existence of a subgraph (corresponding to delete operations) epimorphism (i.e. surjective homomorphism, corresponding to merge operations) from a source graph to a reduced graph. Subgraph epimorphisms (SEPI) differ from subgraph isomorphisms (SISO) by allowing merge operations in addition to delete operations. On undirected graphs, SEPIs differ from minors [13] with respect to the three following points: (i) non adjacent vertices may be merged, (ii) merging adjacent vertices creates loops, and (iii) arcs cannot be deleted without deleting or merging vertices.

In this paper, we study the theoretical properties of SEPIs in general directed graphs and relate these properties with other standard notions of graph homomorphisms [9], namely subgraph isomorphisms, minors and graph epimorphisms (EPI).

Main results and overview of the paper. In Section 2, we introduce three partial orders on digraphs respectively based on SEPI, SISO and EPI, and show that, unlike the minor relation, they are not well quasi orders. In Section 3, we introduce three graph distance measures, respectively based on SEPI, SISO and EPI, and we compare them. We show that they are metrics and that these distances are equivalent to graph edit distances defined as the minimum number of edit operations that transform a first graph into another one. In Section 5, we show the NP-completeness of the SEPI existence problem. In Section 6, we present a constraint satisfaction algorithm that has been successfully used to solve practical SEPI problems on a large benchmark of reaction graphs from systems biology. In Section 7, we discuss extensions to non directed and bipartite graphs.

2. Partial Order Relations SISO, EPI and SEPI

2.1. Notations and Definitions

A directed graph, or graph for short in this paper, is a pair (V, A) where V is a finite set of vertices and $A \subseteq V \times V$ a set of arcs. The cardinality of a set S is denoted $|S|$. The size $|G|$ of a graph $G = (V, A)$ is its number of vertices, $|G| = |V|$.

For the remainder of this section, G and G' denote graphs, with $G = (V, A)$ and $G' = (V', A')$.

Definition 2.1 (Graph isomorphism)**.** *An isomorphism from G to G' is a bijective function $f : V \to V'$ such that $(u, v) \in A$ iff $(f(u), f(v)) \in A'$.*

Two graphs G and G' are isomorphic when there exists a graph isomorphism from G to G'. Graph isomorphism is an equivalence relation on directed graphs: we note $\mathcal{G}$ the set of all graphs quotiented by this equivalence relation.

Definition 2.2 (Graph epimorphism)**.** *An epimorphism (*EPI*) from G to G' is a surjective function $f : V \to V'$ such that*

- *for all $u, v \in V$, if $(u, v) \in A$, then $(f(u), f(v)) \in A'$ (graph homomorphism), and,*
- *for all $(u', v') \in A'$, there exists $(u, v) \in A$ such that $f(u) = u'$ and $f(v) = v'$ (surjectivity on arcs).*

If f is bijective, then f is a graph isomorphism. Graph epimorphisms relax the bijection constraint of graph isomorphisms to a surjection constraint on both vertices and arcs (hence the terminology of epimorphism) so that several vertices of G may be mapped on a same vertex of G'. Graph epimorphisms are closely related to graph compactions: on the class of irreflexive graphs (graphs without loops), graph epimorphisms are actually equivalent to graph compactions [19]. Graph epimorphisms are also closely related to quotient graphs (see Section 4.3).

Definition 2.3 (Induced subgraph)**.** *Let $U \subseteq V$ be a subset of vertices of G. The subgraph of G induced by U is $G_{\downarrow U} = (U, A \cap (U \times U))$.*

Definition 2.4 (Subgraph isomorphism)**.** *A* subgraph isomorphism (SISO) *from G to G' is an isomorphism f from an induced subgraph G_0 of G to G'.*

G_0 is the domain of f, denoted by $\operatorname{dom} f$*.*

Definition 2.5 (Subgraph epimorphism)**.** *A* subgraph epimorphism (SEPI) *from G to G' is an epimorphism f from an induced subgraph G_0 of G to G'.*

G_0 is also denoted $\operatorname{dom} f$*.*

Example 1. *The two graphs given in the introduction for the reduction of Michaelis-Menten are related by a* SEPI *where the induced subgraph of the first graph is obtained by deleting the vertices ES and d, and where both vertices c and p are mapped to the vertex c of the second graph.*

These notions thus define three relations over directed graphs, we write

- $G \overset{\text{EPI}}{\rightsquigarrow} G'$ if there exists a graph epimorphism from G to G';
- $G \overset{\text{SEPI}}{\rightsquigarrow} G'$ if there exists a subgraph epimorphism from G to G';
- $G \overset{\text{SISO}}{\rightsquigarrow} G'$ if there exists a subgraph isomorphism from G to G'.

One can easily check that the three relations $\overset{\text{SEPI}}{\rightsquigarrow}$, $\overset{\text{EPI}}{\rightsquigarrow}$ and $\overset{\text{SISO}}{\rightsquigarrow}$ are partial orders over $\mathcal{G}$.

2.2. Morphisms and Graph Operations

These relations are also closely related to graph transformations by delete and/or merge operations. The *delete* operation removes a vertex v from a graph G, together with every arc incident to v. In other words, it reduces G to the subgraph induced by all vertices but v.

Definition 2.6 (Delete)**.** *Let $u \in V$. The result of the deletion of u in G is the induced subgraph $d_u(G) = G_{\downarrow V \setminus \{u\}}$.*

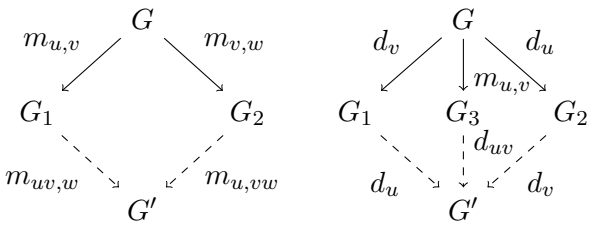
We write $G \to_d G'$ whenever $\exists u, G' = d_u(G)$.

The *merge* operation removes two vertices from a graph and replaces them with a new one inheriting all incident arcs.

Definition 2.7 (Merge)**.** *Let $u, v \in V$ such that $u \neq v$, and let uv be a new symbol such that $uv \notin V$. The result of the merge of u and v in G is the graph $m_{u,v}(G)=(V', A')$ such that $V' = V \setminus \{u, v\} \cup \{uv\}$ and $A' = A \cap (V' \times V') \cup \{(uv, w) \mid (u, w) \in A \text{ or } (v, w) \in A\} \cup \{(w, uv) \mid (w, u) \in A \text{ or } (w, v) \in A\}$.*

We write $G \to_m G'$ whenever $\exists u, v, G' = m_{u,v}(G)$.

We have shown in [8] that these graph operations enjoy the following commutation and association properties:



These permutation properties establish the equivalence between the existence of a (sub)epimorphism from one graph to another one and the existence a finite sequence of delete and/or merge operations leading from the first graph to the second. This is also true for subgraph isomorphisms:

Definition 2.8. *We write $G \to_{md} G'$ if $G \to_d G'$ or $G \to_m G'$.*

Let $o \in \{m, d, md\}$. We write $G' \leftarrow_o G$ if $G \to_o G'$, whenever it is convenient.

We write $G_1 \mathcal{R} G_2 \mathcal{R} G_3$ if $G_1 \mathcal{R} G_2 \wedge G_2 \mathcal{R} G_3$.

We write $G \mathcal{R}^* G'$ whenever there is a string of $\mathcal{R}$ relations from G to G', i.e whenever $G = G'$ or $\exists G_1 \in \mathcal{G}$ s.t. $G \mathcal{R} G_1 \mathcal{R}^* G'$.

Theorem 1 ([8])**.** *$G \overset{\text{EPI}}{\rightsquigarrow} G'$ if and only if $G \to_m^* G'$.*

$G \overset{\text{SEPI}}{\rightsquigarrow} G'$ if and only if $G \to_{md}^* G'$.

$G \overset{\text{SISO}}{\rightsquigarrow} G'$ if and only if $G \to_d^* G'$.

2.3. Properties

SEPI is related to both EPI and SISO since graph epimorphisms and subgraph isomorphisms are subgraph epimorphisms, *i.e.*, $(\overset{\text{EPI}}{\rightsquigarrow} \cup \overset{\text{SISO}}{\rightsquigarrow}) \subseteq \overset{\text{SEPI}}{\rightsquigarrow}$

Hereditary properties have been widely studied for SISO and there exist many properties that are preserved by vertex deletions [2, 3]. However, most of these properties are not preserved when considering both delete and merge operations. This comes from the fact that not only vertex mergings preserve less properties than vertex deletions, but the properties preserved by the two operations are often incompatible.

Nevertheless, one can easily check that SEPIs preserve a few graph properties.

Proposition 2. *Graph completeness is preserved by SEPIs.*

Proposition 3. *Arc symmetry is preserved by SEPIs.*

This proposition shows that SEPIs are well defined on undirected graphs.

Definition 2.9 (Non-neighbors)**.** *The set of outgoing (respectively incoming)* non-neighbors *of a vertex u in a graph G is the set of vertices $ONN(u, G) = \{v \in V \mid (u, v) \notin A\}$ (respectively $INN(u, G) = \{v \in V \mid (v, u) \notin A\}$).*

These non-neighbor sets are monotonic with respect to SEPI:

Proposition 4. *Let f be a* SEPI *from G to G'. Then, for any vertex $x \in \operatorname{dom} f$, $f(ONN(x, G)) \supseteq ONN(f(x), G')$, and $f(INN(x, G)) \supseteq INN(f(x), G')$.*

Proof. If $ONN(x', G') = \emptyset$, the case is immediately proved.

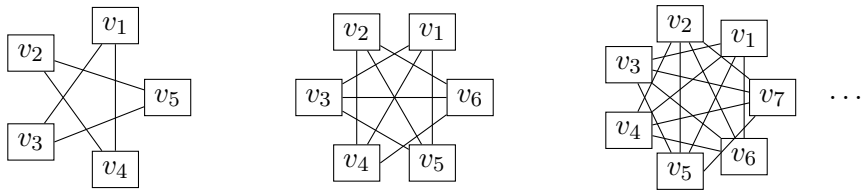
Suppose $ONN(x', G') \neq \emptyset$, and let $y' \in ONN(x', G')$. By surjectivity of f, let y such that $f(y) = y'$. We have $(x', y') \notin A'$, so, since f is a morphism, $(x, y) \notin A$. Thus $y \in ONN(x, G)$, which proves $y' \in f(ONN(x, G))$.

The proof for INN is similar. □

It is worth noting that SEPIs differ from minors in several ways: (i) minors allow the deletion of any arcs whereas SEPIs only allow the deletion of vertices with their adjacent arcs (ii) SEPIs allow the merging of non adjacent vertices whereas minors only allow the merging of adjacent vertices and (iii) SEPIs create loops when merging adjacent vertices, whereas minors don't. Being a graph minor is a partial order over the set of undirected graphs and this partial order is a well-quasi-ordering [16]. This is not the case for SEPI:

Proposition 5. SEPI *is not a Well-Quasi-Order.*

Proof. One can exhibit an infinite antichain of graphs for SEPI. Let $G_n = (V_n, A_n)$ for $n \geq 5$, with $V_n = \{1...n\}$ and $A_n = \{(i, j)$ such that $|i - j| > 1[n]\}$.



This family is an infinite antichain for SEPI. First, one can easily check that $\forall n, \forall u \in V_n, |ONN(x, G_n)| = 3$ (in absence of loop a vertex is in its own ONN set). Let f be a SEPI from G_n to G_m, with $n \geq m$.

Second, f does not delete any vertex. Suppose $D = \{i \mid i \notin \operatorname{dom} f\}$ is not empty. D cannot be V_n either, or G_m would not have any vertex. So there exists i such that one vertex in $\{i, (i + 1)[n]\}$ is deleted and the other is not. Without loss of generality, say i is not deleted and $(i + 1)[n]$ is. Then, by Property 4, $f(i)$ has at most 2 outgoing non-neighbors in G_m: itself and, if defined, the image of $(i - 1)[n]$. This is impossible, since every vertex of G_m has exactly 3 outgoing non-neighbors.

Next, f does not merge any vertices. Suppose f merges i with at least another vertex j. i and j have at most two outgoing non-neighbors in common. Indeed, let us remind that G_n is defined for $n \geq 5$: they have either two outgoing neighbors in common (when $|i - j| = 1[n]$), one (when $|i - j| = 2[n]$), or none (when $|i - j| > 2[n]$). Merging i and j makes them lose their outgoing non-neighbors not in common, thus they lose at least one outgoing non-neighbor, which is once again impossible.

Finally, f must be the identity so that $n = m$. □

Corollary 6. EPI *and* SISO *are not Well-Quasi-Orders.*

Proof. Just notice that an antichain for SEPI is also an antichain for EPI and SISO. □

3. Graph Edit Distance

Each of the three relations introduced in Section 2 may be used to compare some graphs. However, some other graphs cannot be compared as these relations are not total orders.

The distance between two graphs G_1 and G_2 can be defined in two main ways:

(i) in a denotational way, by means of the size of a largest subgraph common to G_1 and G_2;

(ii) in an operational way, by means of the minimum cost sequence of graph edit operations that should be performed to transform G_1 into G_2.

In [4], Bunke has connected these two definitions by introducing a special cost function for the graph edit distance and by showing that under this cost function the graph edit distance problem is equivalent to the maximum common subgraph computation.

While it is always possible to transform one graph into another by performing a sequence of vertex insertion and deletion operations, the size of such a sequence may not be representative of the similarity of the two graphs. Indeed, in some applicative contexts, it is more relevant to measure the distance between two graphs not only by means of vertex insertion and deletion operations but also by means of vertex merge and split operations, thus leading to the extended graph edit distance [1, 5].

In this section, we introduce a graph edit distance corresponding to delete and merge operations. This distance is a simplified case of the extended graph edit distance introduced in [1], which has been defined for labeled graphs and is parameterized by edit costs. In the next section, we shall relate our graph edit distance to SEPI, EPI, and SISO.

Let us first define the edition graphs, which associate a vertex with every graph of $\mathcal{G}$, and an arc with every pair of graphs that can be transformed by applying one operation. Three different edition graphs can be defined according to the different kinds of operation that may be applied, i.e., m (merge), d (delete) or md (merge or delete).

Definition 3.1 (Edition graph)**.** *Let $o \in \{m, d, md\}$. We define the edition graph $\mathcal{E}_o = (\mathcal{G}, \mathcal{A}_o)$ such that $\mathcal{A}_o = \{(G_1, G_2) \in \mathcal{G} \times \mathcal{G} \mid G_1 \to_o G_2\}$.*

These edition graphs are not strongly connected. For example, there is no path from any graph G to any graph G' that has more vertices than G, as for every arc $G_1 \to_o G_2$, we have $|G_2| = |G_1| - 1$. This prevents us from defining a metric using paths. However, there is a natural definition using the *walks* of $\mathcal{E}_o$. A path from s to t only crosses arcs forwards, walks extend paths by allowing to cross arcs forwards and backwards:

Definition 3.2 (Walk of a graph)**.** *Let $G = (V, A)$ be a graph, and s, t be vertices of G. A* walk *w from s to t is a finite sequence $w = (a_1 \dots a_n)$ of arcs of A such that $\exists x_0 \dots x_n \in V, x_0 = s, x_n = t$, and $\forall i, 1 \leq i \leq n, a_i = (x_{i-1}, x_i) \vee a_i = (x_i, x_{i-1})$.*

The length *of w is $|w| = n$.*

Let us now define a graph edit distance as the length of a shortest walk.

Definition 3.3 (Distance)**.** *Let $o \in \{m, d, md\}$, and $G_1, G_2 \in \mathcal{G}$. The distance $d_o : \mathcal{G} \to \mathbb{N} \cup \{+\infty\}$ is*

$$\begin{array}{lcll} d_o(G_1, G_2) & = & \min\{|w| \text{ s.t. } w \text{ is a walk of } \mathcal{E}_o \text{ from } G_1 \text{ to } G_2\} & \text{if a walk exists} \\ d_o(G_1, G_2) & = & +\infty & \text{otherwise.} \end{array}$$

Example 2. *On the graphs G and G' of the Michaelis-Menten reduction given in the introduction, we have*

$$\begin{aligned} d_{md}(G,G') &= 3 \\ d_m(G,G') &= 3 \\ d_d(G,G') &= 5 \end{aligned}$$

For $o \in \{d, md\}$, the distance $d_o(G,G')$ is never $+\infty$, as there always exists a walk from G to G' that goes through the empty graph. However, when $o = m$, it may happen that $d_o(G,G') = +\infty$. This is the case, for example, when G has no arcs whereas G' has at least one arc.

One can easily check that d_o is a metric on $\mathcal{G}$ as it satisfies the non-negativity, symmetry, separability and triangular inequality properties.

Proposition 7. *Let $G, G' \in \mathcal{G}$. Then:*

$$\begin{aligned} d_{md}(G,G') &\leq d_m(G,G') \\ d_{md}(G,G') &\leq d_d(G,G') \\ d_m(G,G') &\leq 3d_d(G,G') \end{aligned}$$

Proof. The two first inequalities can be proved using $\mathcal{E}_{md} \supseteq \mathcal{E}_m$ and $\mathcal{E}_{md} \supseteq \mathcal{E}_d$. The third can be proved by simulating every merge operation by the deletion of both vertices to be merged, and addition (undeletion) of the merged vertex. □

4. Relationship between d_o and EPI, SEPI and SISO

In [4], Bunke has shown that the graph edit distance that only considers vertex deletions (i.e., d_o when $o = d$) is related to the size of the maximum common subgraph. In this section, we extend this result to graph edit distances that consider vertex merges (i.e., d_o when $o \in \{m, md\}$) by relating them to EPI and SEPI.

To show this relationship, we show that for any walk w of $\mathcal{E}_o$ from G to G', there always exists a walk w' from G to G' such that $|w| \geq |w'|$ and w' changes directions at most once, i.e., it first only crosses arcs forward and then only crosses arcs backward. The vertex of $\mathcal{E}_o$ that separates forwards and backwards arc crossings corresponds to an upper bound of G and G' with respect to the partial ordering relations EPI, SEPI, or SISO.

Now, how can one compute the distance d_o between two graphs G, G'?

If $o \in \{d, md\}$, the simplest walk through the empty graph is of size $|G| + |G'|$, $d_o(G,G') \leq |G| + |G'|$, so it is sufficient to explore the graphs from size 0 to $|G| + |G'|$. When $o = d$, one can actually bound the search to graphs of the same size as $|G| + |G'|$.

It is however possible to restrict the exploration of walks to those walks that change directions at most once, by first only going down arcs and then only going up. In order to show this, we introduce quotients of graphs by equivalence relations.

4.1. Preliminaries on Equivalence Relations

In this section, S is a finite set. A binary relation α over S is called an *equivalence relation* over S iff it has the following properties:

- reflexivity : $\forall x \in S, (x,x) \in \alpha$
- symmetry : $\forall x \in S, \forall y \in S, (x,y) \in \alpha \Rightarrow (y,x) \in \alpha$
- transitivity : $\forall x \in S, \forall y \in S, \forall z \in S, (x,y) \in \alpha \wedge (y,z) \in \alpha \Rightarrow (x,z) \in \alpha$

Definition 4.1. *Let α be an equivalence relation over a set S and $x \in S$. The class of x modulo α, denoted $[x]_\alpha$, is $[x]_\alpha = \{y \in A \mid (x, y) \in \alpha\}$.*

The set of classes modulo α, denoted S/α, is $S/\alpha = \{[x]_\alpha \mid x \in S\}$.

Definition 4.2 (Transitive closure)**.** *Let $\alpha \subseteq X \times Y, \beta \subseteq Y \times Z$. The* composition *of α and β is $\alpha \cdot \beta = \{(x, z) \mid \exists y, (x, y) \in \alpha \wedge (y, z) \in \beta\}$.*

The transitive closure *of α is the relation $\alpha^+ = \cup_{i=1}^{\infty} \alpha^i$ with $\alpha^1 = \alpha$ and $\forall i \geq 2, \alpha^{i+1} = \alpha \cdot \alpha^i$.*

The reflexive transitive symmetric closure *of α is the relation*

$$\alpha^{\equiv} = \{(x, y) \mid (x, y) \in \alpha \vee (y, x) \in \alpha \vee x = y\}^+.$$

For any α, $\alpha^{\equiv}$ is an equivalence relation, the smallest containing α.

Definition 4.3. *Let α, β be equivalence relations over S. The* product of equivalence relations *is $\alpha * \beta = (\alpha \cup \beta)^+$. It is an equivalence relation, the smallest (inclusion-wise) containing both α and β.*

4.2. Dimension of an Equivalence Class

For the remainder of this section, let α and β be equivalence classes over S.

Definition 4.4. *Let s be a binary relation over S.*

s is called a spanning *of α iff $s^{\equiv} = \alpha$.*

s is called a free family *(or just* free *for short) iff it has no loops and no cycles.*

As subsets of $S \times S$, spannings are ordered by inclusion. The minimal spannings are free, analogously to minimal generating families in vector spaces. Minimal spannings share a common size, which enables the definition of dimension:

Proposition 8. *Let s be a spanning of α. Then s is minimal iff s is free.*

In this case, $|s| = |E| - |E/\alpha|$.

Proof. Suppose s is minimal and has a cycle $e_1 \dots e_n$. Since $e_n \in (s - \{e_n\})^{\equiv}$, $s - \{e_n\}$ is a spanning of α, so s is not minimal, which is absurd. Likewise, s has no loops.

Now suppose s is free. Let us prove $|s| = |E| - |E/\alpha|$. Let $n = |E/\alpha|$. Notice that $n \geq 1$. When $n = 1$, the undirected version of s ($s \cup s^{-1}$) is a tree, so if it covers $k \geq 1$ vertices, it has $k - 1$ arcs.

We have $s \subseteq \bigcup_{[x]_\alpha \in E/\alpha} [x]_\alpha \times [x]_\alpha$, so $s = \bigcup_{[x]_\alpha \in E/\alpha} s \cap ([x]_\alpha \times [x]_\alpha)$.

Since $s \cap ([x]_\alpha \times [x]_\alpha)$ is a free spanning of $[x]_\alpha \times [x]_\alpha$, the argument for $n = 1$ gives $|s \cap ([x]_\alpha \times [x]_\alpha)| = |[x]_\alpha| - 1$.

So $|s| = \sum_{[x]_\alpha \in E/\alpha} (|[x]_\alpha| - 1) = |E| - |E/\alpha|$.

To conclude, every free spanning has cardinality $|E| - |E/\alpha|$. If $s_0 \subseteq s$ with s_0 minimal, since s_0 is also free, $|s_0| = |s|$, so $s = s_0$ and s minimal. □

From Property 8, one can define the dimension of an equivalence relation as follows:

Definition 4.5. *The* dimension *$dim(\alpha)$ of α is the the size of its minimal spannings $|E| - |E/\alpha|$.*

One can then show a theorem analogous to the incomplete basis theorem:

Theorem 9. *Let $s \subseteq \alpha$ be a free family. Then there is a minimal spanning t of α that contains s.*

Proof. Let us build an increasing sequence of free families of α that contains s. Let $s_0 = s$. If s_n doesn't span α, then for any $(x_n, y_n) \in \alpha - s_n^{\equiv}$, $s_n \cup \{(x_n, y_n)\}$ is free. In this case we define $s_{n+1} = s_n \cup \{(x_n, y_n)\}$. This sequence grows strictly within a finite set, so it has to stop at some m. Then s_m has to span α. Since s_m is free, it is minimal. □

Next, we show that maximal free families are generating families.

Proposition 10. *Let $s \subseteq \alpha$ be free.*
If s is a maximal free family, then s is a spanning of α.
If s has size $dim(\alpha)$, then s is a spanning of α.

Proof. For the first assertion, suppose s doesn't span α, that is, $\exists (x, y) \in \alpha - s^{\equiv}$. Then $x \neq y$, since $s^{\equiv}$ is reflexive. Since $s \cup \{(x, y)\}$ is not free, it must have a cycle that must contain (x, y), with every other arc of the cycle in s. This last statement means that $(x, y) \in s^{\equiv}$, which is absurd.

For the last assertion, suppose free family $s \subseteq \alpha$ has size $dim(\alpha)$. Using Theorem 9, let t be a minimal spanning of α that contains s. Since $dim(\alpha) = |t| \geq |s| = dim(\alpha)$, we have $s = t$. □

Proposition 11.

$$dim(\alpha * \beta) \leq dim(\alpha) + dim(\beta)$$

Proof. Let s, t be minimal spannings of α, β. $s \cup t$ is a spanning of $\alpha * \beta$, with $|s \cup t| \leq |s| + |t|$; a minimal spanning will be even smaller. □

The equality case gives an interesting result: in this case, for every s, t minimal spannings of α, β, $s \cup t$ has no cycle.

4.3. Quotients of Graphs by Equivalence Relations

In this section, $G = (V, A)$ is a directed graph (i.e. $A \subseteq V \times V$), and α, β are equivalence classes over V.

Definition 4.6 (Quotient Graph)**.** *The* quotient *of G by α is:*
$G/\alpha = (\{[x]_\alpha \mid x \in V\}, \{([x]_\alpha, [y]_\alpha) \mid (x, y) \in A\})$.

Notice that G/α has $|V/\alpha|$ vertices.

Graph epimorphisms and graph quotients are strongly linked, in the sense conveyed by the following theorem:

Theorem 12. *There exists an epimorphism f from G to G' iff there exists an equivalence relation α over V such that G/α is isomorphic to G'.*

Proof. $\Rightarrow$ Let $\alpha = \{(x, y) \in V^2 \mid f(x) = f(y)\}$. Then G/α is isomorphic to G'.

$\Leftarrow$ Let $f : V \to \mathcal{P}(V)$ such that $\forall u \in V, f(u) = [u]_\alpha$. Then f is an epimorphism from G to G/α.

□

We shall use a classical theorem about equivalence classes.

Theorem 13. *Let γ be an equivalence relation with $\alpha \subseteq \gamma$. Then γ/α is an equivalence relation over V/α, and $dim(\gamma/\alpha) = dim(\gamma) - dim(\alpha)$.*

Proof. Showing that γ/α is an equivalence class over V/α is easy. The result on dimensions is less well-known.

Let s be a minimal spanning of α. From Property 8 s is a free family of γ, so by Property 9, there is a $t \supseteq s$ that is a minimal spanning of γ. t being a graph, we can consider quotienting it by α. One can show that t/α is actually a spanning of γ/α.

Now t/α generally contains loops and may not be a minimal spanning. Let $t_0 = t - s$. The arcs s/α are exactly the loops of t/α, which means $(t_0/\alpha)^{\equiv} = (t/\alpha)^{\equiv} = \gamma/\alpha$

One can prove that t_0/α is free, and its size is $|t| - |s|$, which allows us to conclude on the dimension of γ/α. □

Proposition 14. *Let γ an equivalence relation such that $\alpha \subseteq \gamma$. Then $(G/\alpha)/(\gamma/\alpha)$ is isomorphic to G/γ.*

Proof. Notice how the vertices of G/γ are subsets of V, and the vertices of $(G/\alpha)/(\gamma/\alpha)$ are subsets of subsets of V. It is easy to check that $m : X \in V/\alpha \longrightarrow \bigcup_{x \in X} x$ is an isomorphism from the latter graph to the former. □

4.4. Graph Distances and Homomorphisms

We now show a practical way to compute distances d_o by restricting the search space.

Let us begin by a simple property of our operations.

Proposition 15. *If there is a sequence of* merge *and* delete *operations that transforms G into G', then this sequence has $|G| - |G'|$ operations.*

Proof. Each *merge* or *delete* operation decrements the number of vertices of G by 1. □

Let us show that in $\mathcal{E}_d$, there is always a short walk with a down-up pattern.

Proposition 16. *Let $G, G_1 = (V_1, A_1), G_2 = (V_2, A_2) \in \mathcal{G}$. If there is a walk $w = G_1 \leftarrow_d^* G \rightarrow_d^* G_2$, then there exists a graph G' and a walk $w' = G_1 \rightarrow_d^* G' \leftarrow_d^* G_2$, with $|w'| \leq |w|$.*

Proof. Let $V' = V_1 \cap V_2$, and $G' = G_{\downarrow V'}$. There is a deletion string from G_1 (respectively G_2) to G', by deleting vertices $V_1 \setminus V_2$ (respectively $V_2 \setminus V_1$).

G contains vertices $V_1 \cup V_2$, but it also has the vertices that have been deleted on both paths to G_1 and to G_2, so that $|G| \geq |V_1 \cup V_2|$.

Using Property 15, w has length $|G| - |G_1| + |G| - |G_2| \geq |V_1 \cup V_2| - |V_1| + |V_1 \cup V_2| - |V_2| = |V_1 \cup V_2| - |V_1 \cap V_2|$, and w' has length $|V_1| - |V_1 \cap V_2| + |V_2| - |V_1 \cap V_2| = |V_1 \cup V_2| - |V_1 \cap V_2|$.

Thus $|w'| \leq |w|$. □

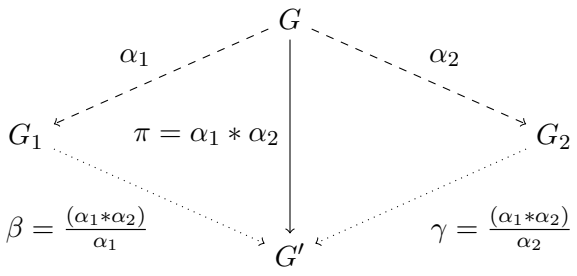
Theorem 17. *If there is a walk w of $\mathcal{E}_d$ from G_1 to G_2, then there is a graph G_c and a walk $w' = G_1 \rightarrow_d^* G_c \leftarrow_d^* G_2$ not longer than w.*

Proof. By recursion on the number of maximal 'peaks' of w, i.e. the number of maximal subwords of $w \in \rightarrow_d \cdot \leftarrow_d$: using Property 16 decreases the number of maximal peaks, and at each step the resulting walk from G_1 to G_2 is shorter or has the same length. □

To show the same kind of properties for merge operations, graph quotients come into play.

Proposition 18. *Let $G, G_1, G_2 \in \mathcal{G}$. If there is a walk $w = G_1 \leftarrow_m^* G \rightarrow_m^* G_2$, then there exists a graph G' and a walk $w' = G_1 \rightarrow_m^* G' \leftarrow_m^* G_2$, with $|w'| \leq |w|$.*

Proof. This illustrates the construction:



Using Theorem 12, we see a string of merge operations as one graph quotient. Let α_1 and α_2 such that $G_1 = G/\alpha_1$ and $G_2 = G/\alpha_2$. Let $\pi = \alpha_1 * \alpha_2$.

With $\beta = \pi/\alpha_1$ and $\gamma = \pi/\alpha_2$, we have, using Property 14, $G/\pi = G_1/\beta = G_2/\gamma$ (in other words, $\frac{G}{\alpha_1 * \alpha_2} = \frac{G}{\alpha_1}/\frac{(\alpha_1 * \alpha_2)}{\alpha_1} = \frac{G}{\alpha_2}/\frac{(\alpha_1 * \alpha_2)}{\alpha_2}$).

So, by transitivity, G' has epimorphisms from G_1 and G_3. And since β and γ have respectively smaller dimensions than α_1 and α_2 (by Properties 11 and 13), the dotted lines in the figure above are shorter than the dashed lines, hence the property.

□

Theorem 19. *If there is a walk w of $\mathcal{E}_m$ from G_1 to G_2, then there is a graph G_c and a walk $w' = G_1 \to_m^* G_c \leftarrow_m^* G_2$ not longer than w.*

Proof. Same proof as Theorem 17, using Property 18. □

This result works for the merge operation. However, it can be extended to the merge and delete operations at the same time. In short, vertex deletion can be simulated by merging with a dummy vertex.

Definition 4.7 (Dummy vertices, pointed graphs)**.** *Let $\perp$ be a fresh symbol that is not a vertex of any graph in $\mathcal{G}$.*

Let $\cdot_\perp : G \in \mathcal{G} \longrightarrow G_\perp = (V_\perp, A_\perp)$, where $V_\perp = V \uplus \perp$ and $A_\perp = A \uplus (V \times \{\perp\}) \uplus (\{\perp\} \times V) \uplus \{(\perp, \perp)\}$.

We call dummy vertex *of a graph G one that has all possible arcs to/from the other vertices of G and to itself. In $G_\perp$, $\perp$ is always a dummy vertex.*

We write the set of pointed graphs $\mathcal{G}_\perp = \{G_\perp \mid G \in \mathcal{G}\}$. We extend the merge operation on $\mathcal{G}_\perp$ with no special treatment for $\perp$: a priori, the image of $\perp$ can be any vertex, and the antecedents of $\perp$ can be any vertex.

Proposition 20. *$\cdot_\perp$ is an isomorphism from $\mathcal{E}_m$ to $(\mathcal{E}_\perp)_m$:*
$G \to_{md} G'$ if and only if $G_\perp \to_m G'_\perp$.

Proof. The left to right implication is straightforward. Let $\mu : G \longrightarrow G'$ the function corresponding to the operation, merging or deletion. If the operation is a merging, then sending $\perp$ to $\perp$ is valid. If it is a deletion, sending $\perp$ to $\perp$ and sending the deleted vertex to $\perp$ makes the operation a merging. So $\mu_\perp : \begin{pmatrix} v \in \mathrm{dom}(\mu) & \longrightarrow & \mu(v) \\ v \in V - \mathrm{dom}(\mu) & \longrightarrow & \perp \\ \perp & \longrightarrow & \perp \end{pmatrix}$ is a merging from $G_\perp$ to $G'_\perp$.

The converse implication should be done carefully. Let us call $\mu_\perp : G_\perp \longrightarrow G'_\perp$ the function corresponding to the merging. Notice that $\mu_\perp(\perp)$ is not necessarily $\perp$. However $\mu_\perp(\perp)$ is necessarily a dummy vertex, so that $\omega : \begin{pmatrix} v \notin \{\mu_\perp(\perp), \perp\} & \longrightarrow & v \\ \mu_\perp(\perp) & \longrightarrow & \perp \\ \perp & \longrightarrow & \mu_\perp(\perp) \end{pmatrix}$ is a graph isomorphism of $G_\perp$.

Let $\rho = \omega \circ \mu_\perp$: it is a merging from $G_\perp$ to $G'_\perp$ with $\rho(\perp) = \perp$. One can check that $\mu : \left(\begin{array}{ccc} v & \longrightarrow & \rho(v) \quad \text{if } \rho(v) \neq \perp \end{array} \right)$ is either a merging (when $\rho^{-1}(\perp) = \{\perp\}$), or a deletion of u (when $\rho^{-1}(\perp) = \{\perp, u\}$). □

Theorem 21. *If there is a walk w of $\mathcal{E}_{md}$ from G_1 to G_2, then there is a graph G_c and a walk $w' = G_1 \rightarrow^*_{md} G_c \leftarrow^*_{md} G_2$ not longer than w.*

Proof. Combining Theorem 19 and Property 20 yields the result. □

These results on down-up walks lead to an interesting corollary when considering morphisms:

Corollary 22. *Let $G, G' \in \mathcal{G}$.*

- $d_d(G, G') = |G| + |G'| - 2\max\{|G_c| \text{ s.t. } G \overset{\text{SISO}}{\rightsquigarrow} G_c \wedge G' \overset{\text{SISO}}{\rightsquigarrow} G_c\}$
- $d_m(G, G') = |G| + |G'| - 2\max\{|G_c| \text{ s.t. } G \overset{\text{EPI}}{\rightsquigarrow} G_c \wedge G' \overset{\text{EPI}}{\rightsquigarrow} G_c\}$
- $d_{md}(G, G') = |G| + |G'| - 2\max\{|G_c| \text{ s.t. } G \overset{\text{SEPI}}{\rightsquigarrow} G_c \wedge G' \overset{\text{SEPI}}{\rightsquigarrow} G_c\}$

Proof. Use Theorem 1 to transpose Theorems 17, 19 and 21 to morphisms, then use Property 15 for cardinalities. □

The first equality is already known: the deletion distance between G and G' is the number of deletions to the maximum common induced subgraph G_c [4], which is *the* greatest lower bound of G and G' for the SISO partial order.

The other two equalities are new: G_c is a greatest lower bound of maximal cardinality for the EPI (respectively SEPI) partial orders. Note that there may be several common graphs of maximum cardinality.

Corollary 23. *$d_o(G, G')$ has the same parity as $|G| + |G'|$.*

5. Computational Complexity

The SISO and EPI decision problems are NP-complete [19], and computing d_{SISO} or d_{EPI} is NP-hard since $d_o(G, G') = |G| - |G'|$ if and only if $G \overset{r}{\rightsquigarrow} G'$, by Proposition 15 and Corollary 22.

Definition 5.1. *The* subgraph epimorphism problem *is the following decision problem:* "given two graphs G and G', is $G \overset{\text{SEPI}}{\rightsquigarrow} G'$ or not ?".

We prove the NP-completeness of the SEPI decision problem by direct reduction of SAT [6]. For a finite set of variables X, let $\neg X = \{\neg x_1, \ldots, \neg x_m\}$ denotes the set of negative literals constructed upon X. For a Boolean formula in conjunctive normal form $\phi = c_1 \wedge \cdots \wedge c_n$ over X, we have for all $1 \leq i \leq n$, $c_i = \ell_{i,1} \vee \cdots \vee \ell_{i,n_i}$, and for all $1 \leq j \leq n_i$, $\ell_{i,j} \in X \cup \neg X$. Let us write $C_\phi = \{i \in \mathbb{N} \mid 1 \leq i \leq n\}$ and $L_\phi = \{(i, j) \in \mathbb{N}^2 \mid 1 \leq i \leq n, 1 \leq j \leq n_i\}$.

Lemma 24. *A Boolean formula ϕ in conjunctive normal form is satisfiable, if and only if there exists a subset $X \subseteq L_\phi$ such that*

- *for all $1 \leq i \leq n$, there exists $1 \leq j \leq n_i$ such that $(i, j) \in X$,*
- *for all $(i, j), (i', j') \in X$, $\ell_{i',j'} \neq \neg \ell_{i,j}$.*

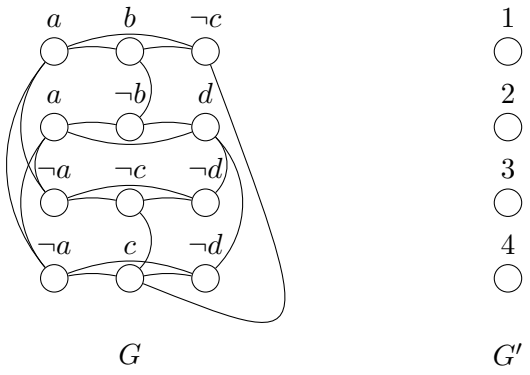


Figure 2: Reduction from the SAT instance $\phi = (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee d) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d)$ to SEPI

Proof. If $\mu : V \to \{0,1\}$ satisfies ϕ, we pose for all $x \in V$, $\mu(\neg x) = 1 - \mu(x)$ and then it suffices to observe that $X = \{(i,j) \in L \mid \mu(\ell_{i,j}) = 1\}$ satisfies the conditions of the lemma. Conversely, given a subset $X \subseteq L_\phi$ satisfying these conditions, we pose $\mu : V \to \{0,1\}$ such that for all $x \in V$, $\mu(x) = 1$ if there exists $(i,j) \in X$ such that $\ell_{i,j} = x$ and 0 otherwise. Then, we observe that μ satisfies ϕ. □

We say that ϕ *is satisfied by* X if X is a subset of L_ϕ satisfying these conditions.

Theorem 25. *The subgraph epimorphism problem is NP-complete.*

Proof. The subgraph epimorphism problem is NP since, given two graphs $G = (V, A)$ and $G' = (V', A')$ and function $f : V \to V'$, checking whether f is a subgraph epimorphism or not can be done in polynomial time. Therefore, it suffices to show that the subgraph epimorphism problem is NP-hard.

Let us assume a SAT instance given by a Boolean formula in conjunctive normal form ϕ over a finite set of variables X. Let G and G' be the following two graphs.

$$G = (L_\phi, \{\{(i,j),(i',j')\} \subseteq L_\phi \mid i = i' \wedge j \neq j' \vee \ell_{i',j'} = \neg\ell_{i,j}\})$$
$$G' = (C_\phi, \emptyset)$$

This construction is depicted in figure 2.

The theorem is then an immediate consequence of the following lemma.

Lemma 26. *ϕ is satisfiable if and only if there exists a* SEPI *from G to G'.*

Suppose that ϕ is satisfied by a set $K \subseteq L_\phi$. Let $K' = \{(i, \min\{j \mid (i,j) \in K\}) \mid 1 \leq i \leq n\}$. K' is a matching, i.e. for all $i \in C_\phi$, there exists a *unique* j_i, with $(i,j_i) \in L_\phi$ such that $(i,j_i) \in K'$. Notice that K' still satisfies ϕ. Let $\mu : (i,j) \in K' \mapsto i \in C_\phi$. We show that μ is a SEPI from G to G'. Indeed, μ is surjective, since K' satisfies ϕ. Furthermore, the subgraph of G induced by K' has no edges: if (i,j) and (i',j') are in K, then either $i = i'$, and then $j = j'$ because K' is a matching, or $i \neq i'$, and then $\ell_{i',j'} \neq \neg\ell_{i,j}$ because $K' \subseteq K$. Since there are no edges to preserve, μ is trivially a morphism. So μ is a SEPI from G to G'.

Conversely, suppose that μ is a SEPI from G to G'. Let $K = \mu^{-1}(C_\phi)$. We show that K satisfies ϕ. First, μ is onto, so $|K| \geq n$. Then, if for some i there are distinct $(i,j),(i,j') \in K$, the edge between (i,j) and (i,j') could not be preserved by μ, so there is at most one $(i,j) \in K$ for each $i \in \{1,\dots n\}$. We deduce $|K| = n$, so there is exactly one $(i,j) \in K$ for each i.

Furthermore, if $\ell_{i',j'} = \neg\ell_{i,j}$ for some (i,j) and $(i',j') \in K$, then the arc between the two vertices could not be preserved by μ. So K satisfies ϕ. □

It is worth noticing that in this proof, the existence of a SEPI from G to G' is equivalent to the existence of a SISO from G to G'. Therefore, this reduction shows the NP-hardness of both SEPI and SISO.

6. Constraint Logic Program

Despite its theoretical computational complexity, SEPI problems can be efficiently solved for some applications. In this section, we present a constraint logic program that has been successfully used to solve practical SEPI problems on a large benchmark of reaction graphs from systems biology [8]. This program implemented in GNU-Prolog [7] uses the built-in constraints `relation` and `element_var` described in the next section.

6.1. Preliminaries on Constraint Satisfaction Problems

Definition 6.1 (CSP)**.** *A* constraint satisfaction problem *(CSP for short) is a triple* $\mathcal{P} = (V, D, C)$*, where:*

- V *is a set of variables*
- D *is a family of domains indexed by variables from* V *:* $\forall X \in V$*,* D_X *is a finite set.*
- C *is a set of constraints, each* $c \in C$ *is defined by its arity* $\mathrm{ar}(c) \in \mathbb{N}$*, a tuple of variables* $\vec{X}(c) \in V^{\mathrm{ar}(c)}$ *and a relation* $R(c) \subseteq \prod_{i=1}^{ar(c)} D_{\vec{X}_i(c)}$*.*

Definition 6.2 (Solution of a CSP)**.** *An assignment* $\eta : X \in V \longrightarrow \eta(X) \in D_X$ *is a solution of* $\mathcal{P}$ *when* $\forall c \in C, (\eta(X_1), \ldots, \eta(X_n)) \in R(c)$*, with* $\vec{X}(c) = (X_1, \ldots, X_n)$*.*

Graph matching problems can be easily modeled as constraint satisfaction problems [12]. One variable per vertex in the source graph is introduced, its domain is the set of vertices of the target graph. Symbolic constraints on these variables are then used to express the matching problem and to actively prune the search space by filtering the domain of these variables during search.

We shall use the following symbolic constraints and their associated domain filtering algorithms (available in GNU-Prolog [7]):

- `relation` constrains variables using a relation given in extension. The constraint $c =$ `relation`$(\vec{Y}, \mathcal{R})$, where $\vec{Y}$ is a tuple of variables and $\mathcal{R}$ is a relation, is defined by:
  - $\mathrm{ar}(c) =$ arity of $\vec{Y}$ = arity of $\mathcal{R} = n$
  - $\vec{X}(c) = \vec{Y}$
  - $R(c) = \mathcal{R} \cap \prod_{i=1}^{n} D_{Y_i}$,
- `element_var` constrains a list of variables f describing a function to have Y as the image of its X-th element. The constraint $c =$ `element_var`(X, f, Y), where $X, Y \in V$ and $f = (f_1, \ldots, f_n) \in V^n$, is defined by
  - $\mathrm{ar}(c) = n + 2$
  - $\vec{X}(c) = (X, f_1, \ldots, f_n, Y)$
  - $R(c) = \{(index, \eta_1, \ldots, \eta_n, image) \in D_X \times (\prod_{i=1}^{n} D_{f_i}) \times D_Y \mid \eta_{index} = image\}$

6.2. SEPI *as a Constraint Satisfaction Problem*

The graph epimorphism problem between two graphs G and G' can be modeled as a constraint satisfaction problem $\mathcal{P} = (V, D, C)$ as follows. Variables are associated to the vertices of G and G' and to the edges of G'

$$V = \{X_v \mid v \in V(G)\} \uplus \{Y_{v'} \mid v' \in V(G')\} \uplus \{Y_{e'} \mid e' \in E(G')\},$$

with domains respectively

$$D(X_v) = V(G'_\perp),$$
$$D(Y_{v'}) = \{1, \ldots, |V(G)|\},$$
$$D(Y_{e'}) = \{1, \ldots, |E(G)|\}.$$

The constraints are

$$\begin{aligned} C = &\{\texttt{relation}((X_u, X_v), E(G'_\perp)) \mid (u, v) \in E(G)\} \\ &\cup \{\texttt{element\_var}(Y_{v'}, X(V(G)), v') \mid v' \in V(G')\} \\ &\cup \{\texttt{element\_var}(Y_{(u',v')}, \pi_1(X(E(G))), u') \mid (u', v') \in E(G')\} \\ &\cup \{\texttt{element\_var}(Y_{(u',v')}, \pi_2(X(E(G))), v') \mid (u', v') \in E(G')\} \end{aligned}$$

where $X(V(G))$ is an arbitrarily ordered list containing the elements of $\{X_v \mid v \in V(G)\}$, $X(E(G))$ is a list representation of the set $\{(X_u, X_v) \mid (u, v) \in E(G)\}$, and π_1, π_2 map the first and second projection functions on lists.

Proposition 27. *The CSP problem $\mathcal{P}$ associated to graphs G, G' has a solution if and only if there exists a subgraph epimorphism from G to G'.*

Proof. We prove that a variable assignment η is a solution to $\mathcal{P}$ if and only if the restriction of η to $\{u \in V(G) \mid \eta(u) \in V(G')\} = \eta|_{V(G)}^{V(G')}$ is a subgraph epimorphism from G to G'.

If η is a solution to $\mathcal{P}$, then $\eta|_{V(G)}^{V(G')}$ preserves the arcs of G since for each $(u, v) \in E(G)$, $\texttt{relation}((X_u, X_v), E(G'))$ enforces $(\eta(X_u), \eta(X_v)) \in E(G')$. Moreover, $\eta|_{V(G)}^{V(G')}$ is surjective on vertices of G', since for each $v' \in V(G')$, $\texttt{element\_var}(Y_{v'}, X(V(G)), v')$ forces the $Y_{v'}$-th element of $V(G)$ to have v' as its image, and similarly surjectivity on arcs of G' is enforced by the remaining constraints.

Conversely, suppose there is a subgraph epimorphism f from G to G'. Let $g : V(G') \mapsto V(G)$ be any inverse of f on the vertices (i.e. $\forall v' \in V(G') f(g(v')) = v'$), and $h : E(G') \mapsto E(G)$ any inverse of f on the arcs. Let η such that:

- $\forall v \in V(G), \eta(X_v) = f(v)$,
- $\forall v' \in V(G'), \eta(Y_{v'}) = i$ s.t. $V(G)_i = g(v')$,
- $\forall (u', v') \in E(G'), \eta(Y_{(u',v')}) = j$ s.t. $E(G)_j = h((u', v'))$.

Then η is a solution to the CSP $\mathcal{P}$. □

Interestingly, the $\perp$ vertices used to reduce SEPI to EPI in the proof of Proposition 20 are also used in this coding of subgraph epimorphism as a CSP: without the dummy vertices, we obtain a CSP encoding of the EPI problem.

6.3. The CSP Framework: Propagation and Enumeration

In order to solve a real-world CSP problem, the enumeration of tuples η is not a viable possibility. Constraint solvers use the fact that from the reduction of the domain of some variables, one can deduce information about the other variables: in particular, one can deduce forbidden values for other variables.

As an example, consider the CSP with variables $\{X, Y, Z\}$ and constraints $(X, Y) \in \{(1, 1), (2, 3)\}$, $(Y, Z) \in \{(1, 2), (1, 3)\}$. Instantiating X to 1 allows the deduction of $Y = 1$, so we never need to test $Y = 2$: this deduction has been *propagated* from $X = 1$. To continue the solving process, we have no choice but to let $Y = 2$, and we can choose the next value to try for Z: $Z = 2$ is a valid choice.

If at first we had tried $X = 2$, the first constraint would have forbidden $Y = 1$, and the second constraint would have forbidden every value of Z. This is a *failure*: on failure, constraint solvers backtrack to the last instantiation $Var = val$ where there was a choice, removes val from the domain of Var, and tries another value for Var. If no value remains, the backtracking process has to return to an even earlier choice point, and if there is no such choice point, then there is no solution to the CSP.

Searching for a solution by constraint solving alternates between propagation and enumeration steps. The general searching scheme is described in the following function $\textsc{Search}_C(D)$: this function returns a solution η in the domain family D if such a solution exists, and fails otherwise.

function $\textsc{Search}_C$(D)
  $D' \leftarrow \textsc{Propagate}_C(D)$
  if $\exists X \mid D'_X = \emptyset$ **then**
    fail
  else if $\forall X, |D'_X| = 1$ **then**
    return η such that $\forall X, D'_X = \{\eta(X)\}$
  else
    $(X, d) \leftarrow \textsc{VariableValueSelection}(D')$
    try
      $D'' \leftarrow (X \mapsto d; Y \neq X \mapsto D'_Y)$
      $\textsc{Search}_C(D'')$
    on failure
      $D'' \leftarrow (X \mapsto D'_X \setminus d; Y \neq X \mapsto D'_Y)$
      $\textsc{Search}_C(D'')$
    end try
  end if
end function

Propagation is assumed to be monotonic ($\textsc{Propagate}_C(D_X) = D'_X \subseteq D_X$ for all $X \in V$) and correct (if the condition $\forall X, |D'_X| = 1$ holds, then the resulting η is a solution). Enumeration is assumed to be strictly monotonic ($d \subsetneq D'_X$) and inhabited ($d \neq \emptyset$). The function VariableValueSelection should never select a singleton variable: $\textsc{VariableValueSelection}(D) = (X, d) \Rightarrow |D_X| > 1$. Therefore, $\textsc{Search}_C(D)$ terminates.

In the worst case, with no propagation ($D' = D$), the search procedure is equivalent to a non-deterministic labeling in $\mathbf{O}(d^n)$ where d is the size of the largest domain (the maximum between the number of vertices in G' and the number of arcs in G) and n the number of variables (the sum of the number of vertices in G and arcs in G').

We consider propagation algorithms that ensures the domain-arc-consistency of the domain family $D' = \textsc{Propagate}_C(D)$ with respect to the set of constraints C. A domain family D' is domain-consistent with respect to a set of constraints C when for every variable $X \in V$ and

every value $v \in D'_X$, there exists an assignment η such that $\eta(X) = v$ and η respects every constraint in C.

The built-in constraint propagators of GNU Prolog indeed maintain arc-consistency for the constraints `relation` and `element_var`.

6.4. Search Strategy for SEPI

Here we discuss the choice of VariableValueSelection for SEPI CSPs.

The default choice could be to use a generic strategy for enumerating both source vertices and antecedent vertices and arcs as follows:

function VariableValueSelection(D)
  if $\exists X \mid D_X = \{d_1, d_2, \ldots\}$ **then**
    return (X, d_1)
  end if
end function

Actually, the enumeration of only one of the sets is sufficient. Let us consider the following enumeration function:

function VariableValueSelection$_{X(V(G))}$(D)
  if $\exists v \in V(G) \mid D_{X_v} = \{d_1, d_2, \ldots\}$ **then**
    return (X_v, d_1)
  else if $\exists v' \in V(G') \mid Y_{v'} = \{d_1, d_2, \ldots\}$ **then**
    return $(Y_{v'}, d_1)$
  else if $\exists (u', v') \in E(G') \mid Y_{(u',v')} = \{d_1, d_2, \ldots\}$ **then**
    return $(Y_{(u',v')}, d_1)$
  end if
end function

Proposition 28. *For a* SEPI *CSP, the search strategy with* `VariableValueSelection`$_{X(V(G))}$ *yields a solution once the source vertex variables are instantiated, for any instantiation of the antecedent variables in their domains.*

Proof. First, suppose we have tried to enumerate the source vertex variables, and failed. Then, the correctness of constraint propagation ensures that there is no SEPI from the source graph to the target graph.

Conversely, if on the contrary the enumeration on source vertex variables succeeded, then there is obviously a morphism. Is it surjective? The domain-arc-consistency of `element_var` removes values $v \in V(G)$ (respectively $(u, v) \in E(G)$) from antecedent variables of v' (respectively (u', v')) as soon as it is known that the image of v (respectively (u, v)) is not v' (respectively (u', v')).

Since every source vertex variable is completely instantiated, the domains of antecedent variables are more than a set of possible antecedents: they are the exact sets of antecedents.

From the Search$_C$ algorithm, every antecedent variable has a non-empty domain, which means the morphism is surjective, and any value for the antecedent variables will satisfy the constraints. □

Let us now consider the enumeration of antecedent variables as follows:

function VariableValueSelection$_{antecedents}$(D)
  if $\exists v' \in V(G') \mid Y_{v'} = \{d_1, d_2, \ldots\}$ **then**
    return $(Y_{v'}, d_1)$
  else if $\exists (u', v') \in E(G') \mid Y_{(u',v')} = \{d_1, d_2, \ldots\}$ **then**
    return $(Y_{(u',v')}, d_1)$

else if $\exists v \in V(G) \mid |D_{X_v}| > 1 \wedge D_{X_v} \neq \{\bot\}$ **then**
return $(X_v, \bot)$
end if
end function

Proposition 29. *For a* SEPI *CSP, the search strategy with* `VariableValueSelection`$_{antecedents}$ *yields a solution once every antecedent variable has been instantiated, by instantiating the remaining non-singleton source vertex variables to* $\bot$.

Proof. First, suppose we have enumerated only the antecedent variables, and failed. Once again, it is obvious that there is no SEPI from the source graph to the target graph.

Conversely, if the enumeration on antecedent variables succeeded, then some source vertex variables already have singleton domains. The induced subgraph formed by the source vertices that correspond to these variables are sufficient to cover G', and the correctness of the `relation` propagator ensures that the variables code a morphism. If we put the $\bot$ value for every remaining source vertex variable, we get a SEPI from the source graph to the target graph. □

An enumeration of the source vertex variables is enough to enforce arc surjectivity. However, compared to enumerating the antecedents variables beforehand, the former strategy checks the surjectivity quite late. Enumerating antecedent variables is sufficient provided we fill the remaining source vertex variables with $\bot$. This "antecedents first" strategy works best in practice.

6.5. Implementation and Evaluation

The preceding constraint satisfaction algorithm can be directly coded in GNU-Prolog [7] using the built-in constraint propagators for the `relation` and `element_var` constraints. Some redundant constraints, such as the `all_different` constraint of [15], the `neighborhood` constraint of [11] or other global constraints [18, 20, 17] have been proposed to improve the domain filtering process on graph matching problems, and in some cases to outperform dedicated algorithms such as Vflib [20, 17]. However, the results reported here concern the constraint logic program described in the previous section without global constraints.

Our benchmark for evaluation comes from the repository of models biomodels.net that is widely used in systems biology. Most of these models are biochemical reaction networks from which one can extract a bipartite reaction graph with 'species' and 'reaction' vertices. The domain constraints ensure that the morphism map species vertices to species vertices, and reaction vertices to reaction vertices.

We have shown in [8] that SEPIs faithfully represent reduction relationships between models of biochemical reaction systems, and that they can be used to relate the models in this repository and organize them in a hierarchy of more or less detailed models.

On the 241 curated models of this repository, 131 are reaction models from which non trivial reaction graphs can be extracted. The average size of the graphs is 56 vertices, with a minimum of 9, a median of 37, and a maximum of 315 vertices.

Our GNU-Prolog program was used to decide the existence of SEPIs in all 131*130 pairs of reaction graphs. Of these 17030 comparisons, 329 were not computed within a time out of 20mn, and 16659 were computed in less than 5 seconds [8].

7. Generalization to Undirected Graphs and Bipartite Graphs

7.1. Undirected Graphs

In undirected graphs, with loops allowed, the definition of SEPI is almost the same, with epimorphisms now preserving adjacency instead of successors.

Property 3 shows that the classical encoding of undirected graphs as symmetric directed graphs helps the conversion to undirected graphs.

The notion of outgoing and incoming non-neighbors can be traded for a notion of non-neighbors. Property 4 can be translated for non-neighbors. The infinite antichain given as proof of Property 5 is symmetric, so it also translates as an infinite antichain of undirected graphs.

The operations we consider in the proofs for distances d_r preserve arc symmetry, which makes the lub characterization of distance work for undirected graphs.

The SEPI existence problem between two undirected graphs is also NP-complete since the proof of Theorem 25 uses symmetric graphs.

7.2. Bipartite Graphs

Our interest in subgraph epimorphisms came from an application in bioinformatics that uses bipartite directed graphs. In this setting, epimorphisms have to preserve bipartition of the vertices. Here, every vertex has one of two labels (say 'circle' and 'square'), vertices with the same label can not be adjacent, and epimorphisms have to preserve these labels.

The infinite antichain in Property 5 can be adapted: instead of taking complete graphs minus maximal cycles, we can consider complete (n, n)-bipartite graphs minus maximal cycles, that is: $G_n = (V_n, A_n)$, with $V_n = \{1 \dots n\} \uplus \{1' \dots n'\}$ and $A_n = \{(i, j'), (j', i) \mid i \neq j'\}$.

The distances are defined the same way. The lub characterizations are the same for d_d and d_m. However, the construction to go from EPI to SEPI needs a slight modification: we add two $\perp$ vertices instead of one, a circle-labeled one and a square-labeled one. The $\perp$ vertex of each type is to be linked with every vertex of the other type. This makes the encoding of SEPI in EPI valid.

Finally, the SEPI decision problem can be proved NP-complete by modifying the proof a little. First, we can suppose there are no clauses that contains both x and $\neg x$ in the SAT instance. We take the same C and L as in the main proof, there are n clauses in the instance.

Then, we modify the source graph $\mathcal{G}_1 = (V_1, A_1)$ by changing cliques to bicliques:

$$\begin{aligned} \mathit{circle}(V_1) &= L \\ \mathit{square}(V_1) &= C \cup B \text{ with } B = \{((i_1, j_1), (i_2, j_2) \in L^2 \mid \ell_{i_1,j_1} = \neg \ell_{i_2,j_2}\} \\ A_1 &= \{(c, l) \in C \times L \mid l = (c, \cdot)\} \\ &\quad \cup \{(l, p) \in L \times B \mid p = (l, \cdot) \vee p = (\cdot, l)\}. \end{aligned}$$

the target graph pattern is then $\mathcal{G}_2 = (V_2, A_2)$

$$\begin{aligned} \mathit{circle}(V_2) &= X' = \{x'_1, \dots x'_n\} \\ \mathit{square}(V_2) &= C' \cup D', \text{ with } C' = \{c'_1, \dots c'_n\} \text{ and } D' = \{d'_1, \dots d'_n\} \\ A_2 &= \{(c'_i, x'_i) \mid 1 \leq i \leq n\} \cup \{(x'_i, d'_i) \mid 1 \leq i \leq n\} \end{aligned}$$

If the SAT instance is satisfiable, deducing a SEPI from $\mathcal{G}_1$ to $\mathcal{G}_2$ is trivial. If there is a SEPI μ from $\mathcal{G}_1$ to $\mathcal{G}_2$, then $\mu^{-1}(X')$ satisfies Lemma 24, concluding the proof.

8. Conclusion

The operations of deleting and merging vertices are natural operations for reducing a graph. While graph reductions through a sequence of vertex deletions (respectively mergings) characterize subgraph isomorphisms (respectively graph epimorphisms), sequences of both vertex deletion and merging operations characterize subgraph epimorphisms. Our proposal is thus to

use subgraph epimorphism for comparing graphs in applications where a more flexible notion than the classical notion of subgraph isomorphism is required.

We have shown that SEPIs preserve graph completeness and arc symmetry and that, just like SISO and EPI, SEPI is not a well quasi order. We have defined the SEPI, EPI and SISO distances between two graphs as the size of the largest SEPI (respectively EPI, SISO) lower bound graphs. These distances are equal to the minimum number of respectively vertex deletion and/or merging operations that are necessary to obtain isomorphic graphs. They are also metrics on graphs, and we have $d_d \geq d_{md}$ and $d_m \geq d_{md}$.

From a computational point of view, we have shown that the existence of a SEPI between two graphs is an NP-complete problem, and have presented a constraint logic program for solving it with good performance in practice on a large benchmark of SEPI model reduction problems from systems biology.

It is worth noticing that, given two graphs G and G', the greatest lower SEPI bounds and the least upper SEPI bounds are also interesting to compute since they represent "intersection" and "union" graphs for the SEPI relation. For instance, in our motivating application in systems biology, these objects correspond to the intersection (respectively union) of models at different levels of details for a given biochemical process. These graphs are not unique but we are confident that the constraint satisfaction algorithm we have presented can be interestingly generalized to compute them.

Acknowledgements

We acknowledge support from the national OSEO BioIntelligence and ANR Calamar (ANR-08-SYSC-003) projects and fruitful discussions with the many participants of these projects.

[1] R. Ambauen, Stefan Fischer, and Horst Bunke. Graph edit distance with node splitting and merging. In *IAPR Workshop on Graph-based Representation in Pattern Recognition*, volume 2726 of *Lecture Notes in Computer Science*, pp. 95–106. Springer-Verlag, 2003.

[2] Béla Bollobás and Andrew G. Thomason. Hereditary and monotone properties of graphs. In R. L. Graham and J. Nesetril, editors, *The mathematics of Paul Erdos, II*, chapter Algorithms and Combinatorics 14, pp. 70–78. Springer-Verlag, Berlin, 1997.

[3] Mieczyslaw Borowiecki and Peter Mihók. Hereditary properties of graphs. In V. R. Kulli, editor, *Advances in Graph Theory*, pp. 42–69. Vishwa International Publications, Gulbarga, 1991.

[4] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[5] Pierre-Antoine Champin and Christine Solnon. Measuring the similarity of labeled graphs. In *ICCBR 2003*, LNAI 2689, pp. 80–95. Springer, 2003.

[6] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pp. 151–158, 1971.

[7] Daniel Diaz. *GNU Prolog user's manual*, 1999–2003.

[8] Steven Gay, Sylvain Soliman, and François Fages. A graphical method for reducing and relating models in systems biology. *Bioinformatics*, 26(18):i575–i581, 2010. special issue ECCB'10.

[9] Pavol Hell and Jaroslav Nesetril. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[10] Ralf Hofestädt. A petri net application to model metabolic processes. *Systems Analysis Modelling Simulation*, 16:113–122, October 1994.

[11] Javier Larrosa and Gabriel Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science*, 12(4):403–422, 2002.

[12] Vianney le Clément, Yves Deville, and Christine Solnon. Constraint-based graph matching. In *15th International Conference on Principles and Practice of Constraint Programming (CP 2009)*, volume 5732 of *Lecture Notes in Computer Science*, pp. 274–288, Lisbon, Portugal, 2009. Springer-Verlag.

[13] László Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86, 2006.

[14] Venkatramana N. Reddy, Michael L. Mavrovouniotis, and Michael N. Liebman. Petri net representations in metabolic pathways. In Lawrence Hunter, David B. Searls, and Jude W. Shavlik, editors, *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pp. 328–336. AAAI Press, 1993.

[15] Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *AAAI 1994*, pp. 362–367, 1994.

[16] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory*, Series B 92(2):325–357, 2004.

[17] Christine Solnon. AllDifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12-13):850–864, August 2010.

[18] Sébastien Sorlin and Christine Solnon. A parametric filtering algorithm for the graph isomorphism problem. *Constraints*, 13(4):518–537, 2008.

[19] Narayan Vikas. *Computational Complexity of Graph Compaction*. PhD thesis, Simon Fraser University, August 1997.

[20] Stéphane Zampelli, Yves Deville, and Christine Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353, 2010.