Solving Subgraph Epimorphism Problems using CLP and SAT

Steven Gay, François Fages, Francesco Santini, Sylvain Soliman

INRIA Paris-Rocquencourt

Abstract. In this work, we compare CLP and SAT solvers on the problem of deciding the existence of a subgraph epimorphism problems between two graphs. Our interest in this variant of graph matching problem stems from the study of model reductions in systems biology. In this setting, model reduction can be formalized as the existence of a sequence of vertex, species or reaction, 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 epimorphism from the first graph to the second. We show how subgraph epimorphism problems can be implemented as CP programs, as boolean clauses, and we compare the two approaches on a large benchmark of reaction graphs from systems biology.

1 Subgraph Epimorphisms

Subgraph epimorphisms (SEPI) can be seen as a variant of subgraph isomorphism (SISO). Our interest in this particular graph relation comes from reaction graphs in systems biology:

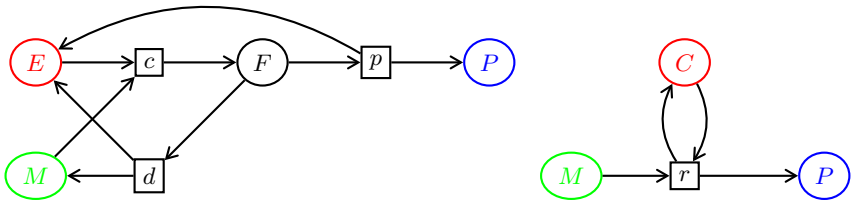

Fig. 1. On the left, an enzymatic mechanism. On the right, the Michaelis-Menten reduced version.

Definition 1 (Graph). *A* graph G *is a pair* $G = (V, A)$*, where* $A \subseteq V \times V$*.*

Definition 2 (Reaction Graph). *A* reaction graph G *is a triple* $G = (V, A, t)$*, where* $t : N \longrightarrow \{s, r\}$ *labels the type of nodes:* $S = t^{-1}(s)$ *is the set of* species *nodes,* $R = t^{-1}(r)$ *is the set of* reaction *nodes, and* $A \subseteq S \times R \cup R \times S$*.*

Example 1. The reaction graph on the left of figure 1 expresses an enzymatic mechanism, usually noted $E + M \leftrightharpoons F \rightarrow E + P$.

The species are represented here by ellipse nodes: $S = \{E, M, F, P\}$. The reactions the rectangle nodes: $R = \{c, d, p\}$. The arcs are $A = \{(M, c), (E, c), (c, F), (d, M), (d, E), (F, d), (p, P), (p, E), (F, p)\}$.

Modelers are interested in having the simplest model that behave as the real-life data, so they apply mathematical reductions to their models.

These reductions induce transformations on the underlying reaction graph, which we intend to capture using graph operations:

Definition 3 (Delete, Merge). *Let $u, v \in V$. The graph $d_v(G)$ is defined as (V', A'), where $V' = V \setminus \{v\}$ and $A' = A \cap (V' \times V')$.*

The graph $m_{u,v}(G)$ is defined as (V', A'), where $V' = V \setminus \{u, v\} \uplus \{uv\}$, $A' = \{(s_{u,v}(x), s_{u,v}(y)) \mid (x, y) \in A\}$, uv is a fresh symbol, and $s_{u,v} : [u \longrightarrow uv, v \longrightarrow uv, x \notin \{u, v\} \longrightarrow x]$.

In the case of reaction graphs, vertices can be merged only if they are both species or both reactions: a reaction can not be merged with a species.

Example 2. In figure 1, take the graph on the left. Delete d and F, then merge c with p. The resulting graph is isomorphic to the graph on the right. This is the classical Michaelis-Menten reduction.

We write $G \rightarrow^*_{md} G'$ when a string of delete and/or merge operation from G yields G'. As strings of delete operation correspond to subgraph isomorphisms, strings of delete/merge operations correspond to subgraph epimorphisms:

Definition 4 (Subgraph Epimorphism). *A subgraph epimorphism from G to G' is a function $f : V \rightarrow V'$ such that $\forall (u, v)$ s.t. $f(u)$ and $f(v)$ defined, $(u, v) \in A \Rightarrow (f(u), f(v)) \in A'$, f surjective (onto) on V' and A'.*

Theorem 1 *There exists a subgraph epimorphism from G to G' iff $G \rightarrow^*_{md} G'$.*

Example 3. In example 2, $m = [M \longrightarrow M, E \longrightarrow C, P \longrightarrow P, c \longrightarrow r, p \longrightarrow r]$.

Deciding SISO is NP-complete, this is also the case for SEPI:

Theorem 2 ([3]) *Deciding the subgraph epimorphism problem is NP-complete.*

This result justifies using approaches such as Constraint Programming and SAT solving to solve SEPI problems.

2 CP Program

In this section, we describe a Constraint Program (CP) to decide the existence of a SEPI from one graph to another.

To differentiate mathematical variables and CP variables, we write CP variables in bold font (as in $\mathbf{X}$ opposed to X) ; $[a, b, c]$ denotes the list of the three elements a, b, c ; π_1 and π_2 are the projection functions, e.g. $\pi_2((a, b)) = b$.

Let G and G' be two graphs, with $G = (V, A)$, $G' = (V', A')$, and $V = \{v_1 \dots v_n\}$, $A = \{a_1 \dots a_k\}$, $V' = \{v'_1 \dots v'_{n'}\}$, $A' = \{a'_1 \dots a'_{k'}\}$, $A'_\perp = A' \cup \{(x, y) \in (V' \cup \{\perp\})^2 | x = \perp \vee y = \perp\} = \{a'_1 \dots a'_{k'}, a'_{k'+1} \dots a'_{k'_\perp}\}$.

2.1 CP Model

The graph epimorphism problem from G to G' can be modeled as a constraint satisfaction problem as follows.

Variables are associated to the vertices and edges of G and G'. Morphism variables: $\mathbf{X}_v$ for $v \in V$, with $D(\mathbf{X}_v) = V' \cup \bot$; $\mathbf{A}_a$ for $a \in A$, with $D(\mathbf{A}_a) = \{1, \ldots, |A'_\bot|\}$. Antecedent variables: $\mathbf{X'}_{v'}$ for $v' \in V'$, with $D(\mathbf{X'}_{v'}) = V$; $\mathbf{A'}_{a'}$ for $a' \in A'$, with $D(\mathbf{A'}_{a'}) = A$.

Constraints to enforce the role of morphism and antecedent variables:

I. Morphism constraints
   i. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_1(a'_1) \ldots \pi_1(a'_{k'_\bot})], \mathbf{X}_{\pi_1(a)})$
   ii. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_2(a'_1) \ldots \pi_2(a'_{k'_\bot})], \mathbf{X}_{\pi_2(a)})$

II. Minimal antecedent constraints
   i. $\forall v \in V, \forall v' \in V', \mathbf{X'}_{v'} = v \Rightarrow \mathbf{X}_v = v'$
   ii. $\forall v \in V, \forall v' \in V', \mathbf{X}_v = v' \Rightarrow \mathbf{X'}_{v'} \leq v$
   iii. $\forall a \in A, \forall a' \in A', \mathbf{A'}_{a'} = a \Rightarrow \mathbf{A}_a = a'$
   iv. $\forall a \in A, \forall a' \in A', \mathbf{A}_a = a' \Rightarrow \mathbf{A'}_{a'} \leq a$

III. Global surjection constraints
   i. $\texttt{gsurjection}([\mathbf{X}_{v_1} \ldots \mathbf{X}_{v_n}], V')$
   ii. $\texttt{gsurjection}([\mathbf{A}_{a_1} \ldots \mathbf{A}_{a_k}], A')$

In order to specialize the CP model to reaction graphs, we can restrict the domains of reaction node variables to reaction nodes and species to species.

Proposition 3 *The CP model $\mathcal{P}$ associated to graphs G, G' has a solution if and only if there exists a subgraph epimorphism from G to G'.*

We use reified constraints and the classical `element` constraint.

We also use a global constraint `gsurjection` that checks that there are more unground variables than uncovered values, with a simple linear time propagator.

While minimal antecedent constraints IIi and IIiii introduce dual variables for surjectivity, constraints IIii and IIiv break symmetries by choosing minimal antecedents. These constraints are redundant with global surjectivity.

2.2 Search Strategy

We tried different search strategies, and the best we found is to enumerate first the $\mathbf{A'}_{a'}$, then the $\mathbf{X'}_{x'}$, and finally the morphism variables.

The following proposition shows that it is not necessary to enumerate on the morphism variables to decide the existence of a SEPI.

Proposition 1. *The SEPI CP above yields a solution iff variables $(\mathbf{X'}_{v'})_{v' \in V'}$ and $(\mathbf{A'}_{a'})_{a' \in A'}$ can be successfully instantiated.*

The proof is elementary and relies on arc-consistency of `element`, see [3].

Therefore in the CP program, the morphism variables need not be instanciated to decide the existence of a SEPI. They are instanciated after the other variables in order to compute a SEPI without further backtracking.

3 SAT model

Coding problems into SAT instances and using a SAT solver to find whether it is satisfiable or not is another successful approach to solve NP complete problems.

A SAT instance can be described as a pair $(X,\ C)$, where X is a set of variables, and C is a set of clauses $c_1 \dots c_r$ with $c_i = \bigvee l_{i,j}$, and finally $l_{i,j}$ is either x or $\bar{x}$, with $x \in X$. A SAT instance can be described more shortly as a boolean formula in conjunctive normal form.

In this section, we will describe, for a given SEPI problem (G, G'), an encoding of the problem into boolean clauses. This encoding has been implemented, and the evaluation will be made in the next section.

The boolean formulae given in this section are transformed into clauses using an obvious normalization procedure : equivalences are broken into two implications, implications $a \to (b \wedge c)$ are broken into $a \to b$ and $a \to c$, implication $a \to b$ is coded in $\neg a \vee b$; no further transformations are done. We write clause(f), where f is a boolean function, to denote the clauses passed to the SAT solver.

We split the description of the coding into two main parts: first how to code a partial surjective function, then how to code a subgraph epimorphism.

3.1 Partial Surjective Function Coding

A SEPI m from G to G' is also a partial surjective function from V to V'.

Definition 5 (Partial Surjective Function). *A binary relation $R \subseteq E \times E'$ is a* partial surjective function *if the following conditions are fulfilled:*

- $\forall x \in E, x'_1 \in E', x'_2 \in E', ((x, x'_1) \in R \wedge (x, x'_2) \in R) \Rightarrow x'_1 = x'_2$
- $\forall x' \in E', \exists x \in E, (x, x') \in R$

The elements $x \in E$ do *not* have to be covered by some $(x, x') \in R$, hence the qualifier *partial*; we write $R(x) = x'$ when $x \in E$ is covered by x', $R(x) = \perp$ when x is not covered.

Variables. We choose a matrix encoding of m seen as a binary relation on $V \times (V' \cup \{\perp\})$. We suppose the elements of $V' \cup \{\perp\}$ are in a total order $v'_0 = \perp < v'_1 < \ldots < v'_{n'}$, and introduce variables: $\forall (v, v') \in V \times (V' \cup \{\perp\}), \mathbf{m}_{v,v'} = 1$ iff $m(v) = v'$; $\forall (v, v') \in V \times (V' \cup \{\perp\}), \mathbf{m}^{<}_{v,v'} = 1$ iff $m(v) < v'$.

Clauses. The following clauses enforce the mathematical description of the variables given in the previous section:

- Left Totality. $\forall v \in V$, i. clause$(\bigvee_{v' \in V' \cup \{\perp\}} \mathbf{m}_{v,v'})$
- Right Totality. $\forall v' \in V'$, i. clause$(\bigvee_{v \in V} \mathbf{m}_{v,v'})$
- Functionality. $\forall (v, v'_j) \in V \times (V' \cup \{\perp\})$,
  i. clause$(\mathbf{m}_{v,v'_j} \Rightarrow \mathbf{m}^{<}_{(v,v'_{j+1})})$
  ii. clause$(\mathbf{m}^{<}_{v,v'_j} \Rightarrow \mathbf{m}^{<}_{(v,v'_{j+1})})$

iii. clause($\mathbf{m}^{<}_{v,v'_j} \Rightarrow \neg \mathbf{m}_{(v,v'_j)}$)

The encoding is self-explaining, except for functionality. Functionality could be encoded by following definition 5. However, this encoding has cubic size: it would have $|V| \cdot |V' \cup \{\bot\}|^2$ clauses. This proves to be a problem in practice.

The coding we provide only has $O(|V| \cdot |V' \cup \{\bot\}|)$ clauses, we achieve this by using the order on $|V' \cup \{\bot\}|$ to force the image of $v \in V$ to be minimal.

3.2 Subgraph Epimorphism Coding

We build on the previous part to constrain the valuation to represent a SEPI.

Variables. We use additional variables: $\forall (a, a') \in A \times A', \mathbf{m}_{a,a'}$ iff $m(a) = a'$ for non deleted arcs ; $\forall a \in A, \mathbf{is\_dummy}(\mathbf{m}_a) = 1$ iff $m(a) = \bot$ for deleted arcs.

Clauses.

- Graph Morphism. $\forall ((u, v), (u', v')) \in A \times A'$,
  - i. clause($\mathbf{m}_{(u,v),(u',v')} \Rightarrow \mathbf{m}_{u,u'}$)
  - ii. clause($\mathbf{m}_{(u,v),(u',v')} \Rightarrow \mathbf{m}_{v,v'}$)
  - iii. clause($(\mathbf{m}_{v,v'} \wedge \mathbf{m}_{v,v'}) \Rightarrow \mathbf{m}_{(u,v),(u',v')}$)
- Subgraph Morphism. $\forall (u, v) \in A$,
  - i. clause($\mathbf{is\_dummy}(\mathbf{m}_{(u,v)}) \Rightarrow \mathbf{m}_{u,\bot} \vee \mathbf{m}_{v,\bot}$)
  - ii. clause($\mathbf{m}_{u,\bot} \Rightarrow \mathbf{is\_dummy}(\mathbf{m}_{(u,v)})$)
  - iii. clause($\mathbf{m}_{v,\bot} \Rightarrow \mathbf{is\_dummy}(\mathbf{m}_{(u,v)})$)
- Left Totality on Arcs. $\forall a \in A$, i. clause($\mathbf{is\_dummy}(\mathbf{m}_a) \vee \bigvee_{a' \in A'} \mathbf{m}_{a,a'}$)
- Right Totality on Arcs. $\forall a' \in A'$, i. clause($\bigvee_{a \in A} \mathbf{m}_{a,a'}$)

Once again the encoding follows the definition closely enough and does not need commentary. The model can be specialized to reaction graphs by restricting domains, i.e. by setting $\mathbf{m}_{v,v'}$ to false when v and v' are not of the same type.

3.3 Surjectivity and Sorting Networks

In order to improve performance a little, we introduce antecedent variables and use boolean sorting networks to constrain surjectivity redundantly.

For a full exposition, see [2] and [1] for the main idea.

4 Performance Evaluation

We implemented the CP model using GNU Prolog 1.4.4, and the SAT model is solved with Glucose 2.2.

To test and compare GNU Prolog and Glucose performance on subgraph epimorphism problems, some of the System Biology models in the *biomodels.net*[1]

[1] `http://biomodels.net`

Table 1. Solvers performance collected in 20min.

Class(Files)	Relations			Nonrelations			Timeouts		
	GNU	Glucose	Union	GNU	Glucose	Union	GNU	Glucose	Union
mapk (110)	38	38	42	60	63	63	12	9	5
circ (110)	17	37	37	60	73	73	33	0	0
caoscill (110)	38	38	38	72	72	72	0	0	0
ccycle (72)	9	12	12	43	51	51	20	9	9

Table 2. Solvers performance collected in 10s.

Class(Files)	Relations			Nonrelations			Timeouts		
	GNU	Glucose	Union	GNU	Glucose	Union	GNU	Glucose	Union
mapk (110)	36	35	41	59	60	60	15	15	9
circ (110)	15	33	33	59	68	68	36	9	9
caoscill (110)	38	38	38	72	72	72	0	0	0
ccycle (72)	9	6	10	42	49	49	21	17	13

repository have been used; in particular, the same models adopted in [4]. A thematic clustering has been accomplished, using information available from the notes of SBML models. The four most populated classes are: *i)* mitogen-activated protein kinase (abbreviated as *mapk*, 11 models), *ii)* circadian clock (*circ*, 11 models), *iii)* calcium oscillations (*caoscill*, 11 models), and *iv)* cell cycle (*ccycle*, 9 models).

Performance has been found on an Intel Core 2 Duo 2.4Ghz processor. The four macro-columns respectively show the number of intra-class comparisons, the number of relations found between models (i.e., of reductions), and the number of no-relations found, and, finally, the number of no-results (where timeout occurs). Each sub-column respectively reports performance for Glucose, GNU Prolog, and the methods combined together, using the same timeout for both. Table 1 shows the results when every computation is limited to 20 minutes, table 2 show the results for a timeout of 10 seconds.

The results show that when our SAT model using Glucose has better performance than the GNU Prolog CLP model.

References

1. Michael Codish and Moshe Zazon-Ivry. Pairwise cardinality networks. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 154–172. Springer, 2010.
2. Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.
3. Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman, and Christine Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 2013. to appear.
4. 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.