Solving Subgraph Epimorphism Problems using CLP and SAT

Steven Gay, François Fages, Francesco Santini, Sylvain Soliman

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 NP-complete 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, species or reaction, deletion and merge operations which 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. We show how subgraph epimorphism problems can be modeled as Boolean constraint satisfaction problems, and we compare CLP and SAT solvers 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:

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 Fig. 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 $A = \{(M, c), (E, c), (c, F), (d, M), (d, E), (F, d), (p, P), (p, E), (F, p)\}$.

When reactions are equipped with kinetics and species with concentrations, it yields a reaction model, which can be simulated. Then simulation can be compared to real-life data, and the model can be modified so that the simulation fits the data, which is the final goal.

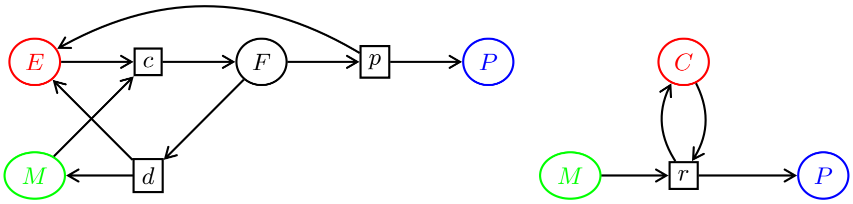

Fig. 1. On the left, an enzymatic mechanism. On the right, the Michaelis-Menten reduced version.

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

These reductions induce transformations on the underlying reaction graph, which can be captured using graph operations:

Example 2. Applying a Michaelis-Menten reduction to the reaction mechanism on the left in Fig. 1 yields the reaction graph on the right, usually written $M + C \to P + C$:

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 3. In Fig. 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.

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

Definition 4 (Subgraph Epimorphism). *A subgraph epimorphism from G to G' is a function $f : V \to V'$ 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 $G \to^*_{md} G'$.*

Example 4. The SEPI corresponding to the example 3 is $m : [M \longrightarrow M, E \longrightarrow C, P \longrightarrow P, c \longrightarrow r, p \longrightarrow r]$.

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

Theorem 2 ([6]) *Deciding the subgraph epimorphism problem is NP-complete.*

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

2 Constraint Program

In this section, we describe how to decide the existence of a SEPI between two graphs using Constraint Programming (CP).

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 first and second projection functions.

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

2.1 CP Model

The existence of a SEPI from G to G' can be modeled using CP as follows.

Variables are associated with the vertices and edges of G and G' :

- Morphism variables
  - $\mathbf{X}_v$ for $v \in V$, with $D(\mathbf{X}_v) = V' \cup \{\perp\}$.
  - $\mathbf{A}_a$ for $a \in A$, with $D(\mathbf{A}_a) = \{1, \dots, |A'_\perp|\}$.
- 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. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_1(a'_1) \dots \pi_1(a'_{k'_\perp})], \mathbf{X}_{\pi_1(a)})$
   ii. $\forall a \in A, \texttt{element}(\mathbf{A}_a, [\pi_2(a'_1) \dots \pi_2(a'_{k'_\perp})], \mathbf{X}_{\pi_2(a)})$

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. $\texttt{gsurjection}([\mathbf{X}_{v_1} \dots \mathbf{X}_{v_n}], V')$
   ii. $\texttt{gsurjection}([\mathbf{A}_{a_1} \dots \mathbf{A}_{a_k}], A')$

This model uses reified constraints and the usual `element` constraint.

It also uses a global constraint `gsurjection` that works as follows. Let $\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_d]$ be a list of variables and $T = [T_1 \dots T_t]$ of sorted list of integers. Let $\text{covered}(\mathbf{D}, T) = \{T_i \mid \exists j, \text{dom}(\mathbf{D}_j) = \{T_i\}\}$ be the elements of T that are taken by some variable, and $\text{committed}(\mathbf{D}, T) = \{\mathbf{D}_j \mid \text{dom}(\mathbf{D}_j) \subseteq \text{covered}(\mathbf{D}, T)\}$ be the variables which can not cover any uncovered variable.

Then $\texttt{gsurjection}(\mathbf{D}, T)$ enforces $|T| - |\text{covered}(T)| \leq |\mathbf{D}| - |\text{committed}(\mathbf{D})|$. When all $\mathbf{D}_j$ are ground, $\mathbf{D}$ has to be a surjection on the elements of T. Implementing a linear time propagator for this constraint is straightforward.

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

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

In order to specialize the CP model to reaction graphs, the domains of reaction (species) node variables can be restricted to reaction (species) nodes.

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 sheds some light on this choice:

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

Proof. Obviously, if enumerating antecedent variables fails, there is no SEPI from the source graph to the target graph.

Conversely, if the enumeration on antecedent variables succeeded, then the corresponding $(\mathbf{X}_v)_{v \in V}$ and $(\mathbf{A}_a)_{a \in A}$ have singleton domains, thanks to domain-arc-consistency of `element` constraints II. The induced subgraph formed by the source vertices and arcs that correspond to these variables are sufficient to cover G', and the morphism constraints I ensure that the variables code a morphism. Giving the $\perp$ value for every remaining morphism variable yields a SEPI from the source graph to the target graph. □

Therefore enumerating morphism variables last ensures there will be no backtracking on these variables: this could explain the good relative performance of this strategy.

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 as a CNF. 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 a CNF using an obvious normalization procedure : implications $a \rightarrow (b \wedge c)$ are broken into $a \rightarrow b$ and $a \rightarrow c$, implication $a \rightarrow b$ 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 adding graph constraints 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* $m \subseteq E \times E'$ *is a* partial surjective function *if the following conditions are fulfilled:*

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

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

Variables. m is encoded as a binary relation on $V \times (V' \cup \{\bot\})$. The elements of $V' \cup \{\bot\}$ are put in a total order $v'_0 = \bot < v'_1 < \ldots < v'_{n'}$.
- $\forall (v, v') \in V \times (V' \cup \{\bot\}), \mathbf{m}_{v,v'} = 1$ iff $m(v) = v'$.
- $\forall (v, v') \in V \times (V' \cup \{\bot\}), \mathbf{m}^{<}_{v,v'} = 1$ iff $m(v) < v'$.

Clauses. The following clauses enforce the mathematical description of the variables:

I. Left Totality. $\forall v \in V$, clause($\bigvee_{v' \in V' \cup \{\bot\}} \mathbf{m}_{v,v'}$)
II. Functionality. $\forall (v, v'_j) \in V \times (V' \cup \{\bot\})$,
    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($\mathbf{m}^{<}_{v,v'_j} \Rightarrow \neg \mathbf{m}_{(v,v'_j)}$)
III. Right Totality. $\forall v' \in V'$, clause($\bigvee_{v \in V} \mathbf{m}_{v,v'}$)

The encoding is self-explanatory, except for functionality. Functionality could be encoded directly as well, with something like:
$\forall (v, v'_1, v'_2) \in V \times (V' \cup \{\bot\})^2$ with $v'_1 \neq v'_2$, clause($\neg(\mathbf{m}_{v,v'_1} \wedge \mathbf{m}_{v,v'_2})$)

This encoding has $|V| \cdot |V' \cup \{\bot\}|^2$ clauses, which is a problem in practice. The coding above, achieved by using the order on $|V' \cup \{\bot\}|$ to force the image of $v \in V$ to be minimal, only has $O(|V| \cdot |V' \cup \{\bot\}|)$ clauses.

3.2 Subgraph Epimorphism Coding

Let us build on the previous part to constrain the function to represent a SEPI.
Variables. Additional variables are used to constrain SEPI:
- Non deleted arcs. $\forall (a, a') \in A \times A', \mathbf{m}_{a,a'}$ iff $m(a) = a'$.
- Deleted arcs. $\forall a \in A, \textbf{is\_dummy}(\mathbf{m}_a) = 1$ iff $m(a) = \bot$

Clauses.
- Left Totality on Arcs. $\forall a \in A$, clause($\textbf{is\_dummy}(\mathbf{m}_a) \vee \bigvee_{a' \in A'} \mathbf{m}_{a,a'}$)
- Right Totality on Arcs. $\forall a' \in A'$, clause($\bigvee_{a \in A} \mathbf{m}_{a,a'}$)
- 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$(\mathbf{is_dummy}(\mathbf{m}_{(u,v)}) \Rightarrow \mathbf{m}_{u,\perp} \vee \mathbf{m}_{v,\perp})$
ii. clause$(\mathbf{m}_{u,\perp} \Rightarrow \mathbf{is_dummy}(\mathbf{m}_{(u,v)}))$
iii. clause$(\mathbf{m}_{v,\perp} \Rightarrow \mathbf{is_dummy}(\mathbf{m}_{(u,v)}))$

Once again the encoding follows the definition closely. 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

The `gsurjection` propagation idea can be imitated with boolean clauses. This improves performance a little. The idea is to introduce minimal antecedents, and then use cardinality networks to force the number of minimal antecedents to be greater than the number of targets.

Cardinality networks use boolean sorting networks to have some consistency using only unit clause propagation.

For a full exposition, [5] compares different approaches to coding integers in boolean clauses, [2] uses such networks on decompositions of cardinality-related constraints, [3] shows that Parberry's odd-even networks behave better than Batcher's merge networks for this purposes, [8] efficiently solves MAXSAT by coding the cardinality part in boolean clauses.

4 Performance Evaluation

We implemented the CLP model using GNU Prolog [4] 1.4.4, and the SAT model is solved with Glucose [1] 2.2.

To test and compare GNU Prolog and Glucose performance on subgraph epimorphism problems, some of the System Biology models in the BioModels repository [9] have been used; in particular, the same models adopted in [7]. 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).

In the experiments reported in Tab. 1, the computation time was limited with a timeout of 20 minutes. Performance has been evaluated 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 (20min + 20min).

Clearly, in order to evaluate the two methods, the interesting value is the number of timeouts: here the efficacy of Glucose can be appreciated, in particular

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

on class *circ* (from 33 no results to nil), but also on class *ccycle* (from 20 no results to 9), and, lastly, a marginal improvement on class *mapk* (from 12 to 9). Class *caoscill* shows no improvement because it is very easy to match even with GNU Prolog (0 no results). These results have been further investigated in detail, discovering that *mapk* is the only class among the four where the GNU Prolog set of relations is not a subset of the one found with Glucose. This difference set (equivalent to $49 \rightarrow 9$, $49 \rightarrow 11$, $49 \rightarrow 28$, $49 \rightarrow 30$) also corresponds exactly to the difference set of no-results between glucose and GNU Prolog. From this the reader can deduce that from a merging of GNU Prolog and SAT implementations, only 4 no-results less on *mapk* can be gained (which would correspond to 4 relations more). Nevertheless, it also possible to deduce that on some models our GNU Prolog version can run more efficiently: in this case, model 49 is better reduced with GNU Prolog than with SAT.

The lists of comparisons for which no result can be obtained with either SAT or GNU are respectively, $\{49 \rightarrow 146, 146 \rightarrow 9, 146 \rightarrow 11, 146 \rightarrow 28, 146 \rightarrow 30\}$ on *mapk*, and $\{56 \rightarrow 7, 56 \rightarrow 111, 56 \rightarrow 144, 109 \rightarrow 7, 109 \rightarrow 111, 109 \rightarrow 144, 144 \rightarrow 111, 144 \rightarrow 169, 144 \rightarrow 196\}$ on *ccycle*. Few of the models seem to represent a bottleneck, due to the high frequency of the same models in these two lists.

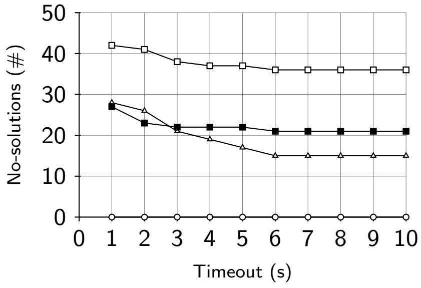

Fig. 2. No-solutions in GNU Prolog.

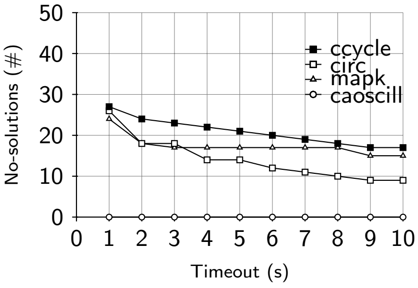

Fig. 3. No-solutions in Glucose.

Moreover, Tab. 2 shows that both implementations also perform well within a short timeout of 10 seconds. This is particularly true with our GNU Prolog implementation (only 7 no-results less over the four classes, from 10sec to 20min), while more debatable with Glucose (23 no-results less in total). Fig. 2 and 3 show how the number of no-solutions decreases by increasing the timeout from 1 up to 10 seconds (GNU Prolog and Glucose respectively).

5 Conclusion

We have compared CLP and SAT approaches for deciding the existence of a subgraph epimorphism from one graph to another.

The main application is to determine the feasibility of computing model reduction hierarchies of real-world model repositories. Section 4 has shown the efficiency of both methods, especially for SAT. In particular, with long timeouts (e.g., 20 minutes), the results found by solving our CNF model with Glucose almost totally subsumes those found by solving out CP model with GNU Prolog.

However, our CLP model can be improved by adding global constraints such as *alldifferent* on antecedents. In addition, we will refine the notion of reduction by adding labels on nodes (e.g., on molecular species), in order to only match related entities; this will also lead to a performance improvement. Moreover, we are currently working on the notion of maximal common subgraph for SEPI, still using both a CLP and a SAT model. This can be used to measure a distance between two biological models.

References

1. Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *IJCAI*, volume 9, pages 399–404, 2009.
2. Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Decompositions of all different, global cardinality and related constraints. In Craig Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09*, pages 419–424, 2009.
3. Michael Codish and Moshe Zazon-Ivry. Pairwise cardinality networks. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 154–172. Springer, 2010.
4. Daniel Diaz, Salvador Abreu, and Philippe Codognet. On the implementation of GNU Prolog. *Theory and Practice of Logic Programming*, 12(1-2):253–282, 2012.
5. Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.
6. Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman, and Christine Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 2013. to appear.
7. 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.
8. Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. Qmaxsat: A partial max-sat solver system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 8:95–100, 2012.
9. Nicolas le Novère et al. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acid Research*, 1(34):D689–D691, January 2006.