Symbolic Methods for Biological Networks D2.1 Report on Scalable Methods for Tropical Solutions (T1.2)

Christoph Lüders, Eléonore Bellot, François Fages, Ovidiu Radulescu, Sylvain Soliman

ANR-DFG SYMBIONT Project ANR-17-CE40-0036 Symbolic Methods for Biological Networks D1.2 Report on Scalable Methods for Tropical Solutions (T1.2)

Christoph Lüders[1], Eléonore Bellot[2], François Fages[2], Ovidiu Radulescu[3], and Sylvain Soliman[2]

[1] *Universität Bonn, Institut für Informatik II, Endenicher Allee 19a, 53115 Bonn, Germany*

[2] *EP Lifeware, Inria Saclay-Île de France, Palaiseau, France*

[3] *DIMNP, UMR CNRS 5235, University of Montpellier, Montpellier, France*

October 7, 2020

Tropical geometry can be used to find the order of time scales of variables in chemical reaction networks and search for model reductions [SGF+15]. In this report, we consider the problem of solving tropical equilibration problems in ODE systems of the BioModels model repository. We are interested in the existence of solutions both in $\mathbb{R}$ and $\mathbb{Z}$. We present three methods and study their scalability to solve complete equilibration problems. The first two methods, a naive polyhedral method using PtCut [Lüd20c], and a Satisfiability Modulo Theories (SMT) method recently introduced in [Lüd20a] using the SMT solver CVC4 [BCD+11], compute the set of solutions over real numbers. The SMT approach is significantly faster than the polyhedral approach, by up to two orders of magnitude. Furthermore, this method provides an anytime algorithm, thus offering a way to compute parts of the solution when the polyhedral approach is infeasible. The third method, the Constraint Programming (CP) method presented in [SFR14] and implemented in Biocham-4, computes integer equilibrations. The CP approach presents similar performance as the SMT method, mostly below two minutes computation time for the polynomial and rational fractional ODE systems in this benchmark. This method also reveals that 30% of the models that can be equilibrated over the reals have in fact no integer solution. These evaluation results show the scalability of the SMT and CP solvers for solving both real and integer tropical equilibration problems on real-size problems.

1. Introduction

Tropical geometry is a natural approach to find the scalings necessary for slow/fast decompositions and model order reduction of biochemical reaction networks [SGF+15]. These reductions extending the quasi-steady state and quasi-equilibrium approximations, well known in biochemistry, are based on automated algorithms. The first step in these algorithms consists in computing tropical equilibration solutions.

The basic scaling idea is to consider that both variables $x_i \in \mathbb{R}_+$ and parameters $k_i \in \mathbb{R}_+$ of the ODE system describing the chemical kinetics are functions of a small, positive parameter. The dominant terms of these functions are considered to be powers of ε. Thus, $x_i = \bar{x}_i \varepsilon^{a_i}$ and $k_i = \bar{k}_i \varepsilon^{\gamma_i}$. For biochemical reaction networks, the reaction rates are rather generally rational functions of the variables x_i, therefore the chemical kinetic equations are rational or polynomial ODEs. Using the variable and parameter scaling, all the monomials terms in these ODEs can be also expressed as powers of ε. Furthermore, the order of the dominant terms in each ODE provide the timescale order of each variable. A scaling satisfies the tropical equilibration conditions when the dominant monomial terms can be compensated, i.e. when not all the dominant monomials have the same sign. A tropical equilibration is complete when the above condition is satisfied for all the ODEs and partial when the above condition applies at least to the ODEs describing the fast variables dynamics. In the latter case, the timescale condition on the variables that are equilibrated has to be solved consistently with the monomial order condition.

In this report, we present three methods and study their scalability to solve complete equilibration problems in the BioModels model repository. The first two methods, a naive polyhedral method using PtCut [Lüd20c], and a Satisfiability Modulo Theories (SMT) method recently introduced in [Lüd20a] using the SMT solver CVC4 [BCD+11], compute the set of solutions over the real numbers.

Satisfiability Modulo Theories (SMT) is usually build on top of SAT (Boolean satisfiability), which was the first problem that was proved, in the form of 3SAT, to be NP-complete. SMT allows to test a logical formula with unknowns and relations for satisfiability and, if it is so, for an assignment of the unknowns that leads to the formula's satisfiability [Mon16]. SMT checking is used today in verification of computer hardware and software and has advanced much in recent years due to advances in technology and industrial applications [DMB11]. The SMT approach introduced in [Lüd20a] and used here is faster than the polyhedral approach for models that would otherwise take more than one minute to compute, in many cases by a factor of 25 or more. Furthermore, this method is an anytime algorithm, thus offering a way to compute parts of the solution when the polyhedral approach is infeasible.

The third method evaluated on the same benchmark here, is the Constraint Programming (CP) method presented in [SFR14] and implemented in Biocham-4. This method searches for integer solutions. The CP approach is similarly efficient as the SMT method, generally under two minutes for the polynomial and rational fractional ODE systems of this benchmark. Interestingly, this method reveals that a large part of the models that can be equilibrated over the reals have in fact no integer solution.

Both SMT and CP methods thus appear to be scalable on BioModels.

The rest of the report is organized as follows. The next section describes the idea of tropical geometry and tropical equilibrations in ODE systems. Section 3 describes the SMT

approach with some details. Section 4 describes the CP approach for searching for integer solutions. Section 5 presents the benchmark of polynomial and fractional ODE systems obtained from the curated part of the BioModels model repository. Then in Section 6, we present the results and compare the performance of the three methods on this benchmark. We conclude in the last section on the scalability of SMT and CP solvers on real and integer tropical equilibration problems for real-size problems and on some possible improvements for future work.

2. Tropical Equilibration Problems

Let $\boldsymbol{x} \in \mathbb{R}_+^d$ and $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathbb{N}_0^m$ be multi-indices with $\boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_i x_i^{\alpha_i}$. We express a polynomial function as

$$P(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{P}} k_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}} - \sum_{\boldsymbol{\beta} \in \mathcal{N}} k_{\boldsymbol{\beta}} \boldsymbol{x}^{\boldsymbol{\beta}},$$

where $\mathcal{P}$ and $\mathcal{N}$ are index sets of the monomials with positive resp. negative sign.

The order of a monomial $k_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}$ is given by the power of ε in that monomial and can be straightforwardly calculated as $\gamma_{\boldsymbol{\alpha}} + \sum_i \alpha_i a_i$, where $\gamma_{\boldsymbol{\alpha}}$ is the order of $k_{\boldsymbol{\alpha}}$ and a_i are the orders of x_i.

When $\epsilon \to 0$ dominance relation between terms is given by their orders, typically $\varepsilon^a << \varepsilon^b$ iff $a > b$. As a consequence, the dominant term is the term of minimal order, and orders are computed according to the rules of min-plus algebra, where multiplication becomes plus and addition becomes min.

The critical observation is that during dynamics on a slow invariant manifold, dominant terms can not remain uncompensated. For the polynomial $P(\boldsymbol{x})$ this means that

$$\min_{\boldsymbol{\alpha} \in \mathcal{P}} \Big(\gamma_{\boldsymbol{\alpha}} + \sum_i \alpha_i a_i \Big) = \min_{\boldsymbol{\beta} \in \mathcal{N}} \Big(\gamma_{\boldsymbol{\beta}} + \sum_i \beta_i a_i \Big), \tag{1}$$

called tropical equilibration condition.

A total tropical equilibration solution satisfies (1) for all the polynomial ODEs or for all numerators for rational ODEs in a common denominator representation.

Observe that (1) comprises now only of minima of linear functions and the set of solutions can thus readily be expressed as a set of polyhedra in the space of variable orders $(a_1, a_2, \ldots, a_n)$.

One polyhedron is defined by each combination of $\boldsymbol{\alpha} \in \mathcal{P}$ and $\boldsymbol{\beta} \in \mathcal{N}$ that yields a hyperplane via

$$\gamma_{\boldsymbol{\alpha}} + \sum_i \alpha_i a_i = \gamma_{\boldsymbol{\beta}} + \sum_i \beta_i x_i, \tag{2}$$

while for all $\boldsymbol{\eta} \in \mathcal{P} \cup \mathcal{N}$ half-spaces are defined by

$$\gamma_{\boldsymbol{\alpha}} + \sum_i \alpha_i a_i \leq \gamma_{\boldsymbol{\eta}} + \sum_i \eta_i a_i. \tag{3}$$

The whole set of polyhedra is defined by cycling over all possible choices for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

Since the ODE system often consists of multiple equations, we get multiple sets of polyhedra. Since all l.h.s. have to be zero, the constraints that define the set of polyhedra all have to hold at the same time, i.e., the sets of polyhedra have to be intersected. The resulting set of polyhedra is called the set of *tropical equilibration solutions*.

3. Solving Tropical Equilibrations Over the Real Numbers

3.1. Satisfyability Modulo Theories (SMT) Method

We are interested in the intersection of several sets of polyhedra. That is, given polyhedra P_{ij} and sets of polyhedra $B_i = \{P_{i1}, P_{i2}, \ldots, P_{in_i}\}$, we are interested in their intersection $\bigcap_i B_i = \{R_1, R_2, \ldots, R_\ell\}$, where the R_i are again polyhedra.

In this article we first show how to use SMT checking to solve this question. Satisfiability Modulo Theories (SMT) allows us to decide if a logical formula, with atoms that are themselves equations or inequalities, is satisfiable or not. For example, $x > 1 \wedge x < 2$ is an SMT formula. One has to specify a *theory* of numbers that unknowns in the formula can assume. In the above example, the problem is satisfiable in the theory of real numbers, but not in the theory of integers. If an SMT problem is satisfiable, SMT can return a *model*, which is an assignment for all unknowns in the formula.

SMT solvers may accept logical formulas in general, but often formulas are written in conjunctive normal form (CNF), i.e., as a number of `AND` clauses called *assertions*. If SMT solvers are used in *incremental mode*, one can, after they have found a solution, add additional assertions and "continue" to look for further solutions, but keep using their internal search heuristics. We make use of that later.

3.2. Polyhedral Representation of the Set of Solutions

A polyhedron is defined as the intersection of finitely many hyperplanes and half-spaces. Furthermore, each hyperplane can be expressed as two (closed) half-spaces, thus a polyhedron can be described as a finite number of half-spaces [Zie95].

Since we are using computers to do the work, we represent numbers $\in \mathbb{R}$ as elements of $\mathbb{Q}$ for reasons of numerical stability. All numbers we are dealing with are either provided with finite precision or can be computed to arbitrary precision.

Hence, given ambient space $\mathbb{Q}^d$, a (closed) half-space H is the set

$$H = \{\boldsymbol{x} \in \mathbb{Q}^d \mid \lambda_0 + \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_d x_d \leq 0, \lambda_i \in \mathbb{Q}\}. \tag{4}$$

Given half-spaces H_k, we define a polyhedron as

$$P = \bigcap_k H_k. \tag{5}$$

A *bag* is what we call a set of polyhedra P_j and describe it as a union, i.e.,

$$B = \bigcup_j P_j.$$

Finally, we are looking for the intersection of said bags B_i, that is

$$V = \bigcap_i B_i = \bigcap_i \bigcup_j P_{ij} = \bigcup_k R_k. \tag{6}$$

The naive solution to the problem of computing the intersection is to cycle successively through all combinations. To do that, pick two bags B_j and $B_{j'}$, $j \neq j'$, and intersect all

polyhedra from one with all polyhedra from the other to form a new bag B'. Then, remove B_j and $B_{j'}$ from the set of bags and insert B' instead. Continue this procedure until there is only one bag left, which will then consist of the sought polyhedra $R_1, R_2, \ldots, R_\ell$. That is the solution that was used in [SGF+15] and, with some refinements, in PtCut [Lüd20c].

The problem with this solution is that the complexity is exponential in the number of bags. In practice, it often happens that the number of intermediate results, i.e., the number of polyhedra in some B', can be very high, even if in the end there are only a few solution polyhedra. This *intermediate expression swell* makes computing the intersection for some models infeasible.

Table 4 shows BioModels from our survey, their number of resulting polyhedra and their maximum number of intermediate polyhedra. More details on the computation can be found in Section 5.

3.3. The SMT Procedure

First, we have to formulate our problem as an SMT problem. Fortunately, it is easy to convert a polyhedron as defined in (5) into a logical formula. Set theory maps easily to logical formulas with union mapping to logical `OR` and intersection to logical `AND`. In the following, $\tilde{H}$ denotes the logical formula that defines the set H.

Thus, (6) expands to

$$\tilde{V} = \bigwedge_i \bigvee_j \bigwedge \tilde{H}_{ijk} \tag{7}$$

and definition (4) of H_{ijk} employs a linear function that can be used as a formula in SMT. Call the resulting SMT formula f. Thus, we can use an SMT solver to decide the satisfiability of formula f and, what's more important, get a point $\boldsymbol{x} \in \mathbb{Q}^d$ that satisfies the constraints.

Next, we are looking for a matching polyhedron that includes point $\boldsymbol{x}$ and is included in the solution V. Since point $\boldsymbol{x}$ is contained in the intersection of the B_i, it must be contained in at least one polyhedron P_{ij} per bag B_i. Thus, we cycle through all B_i to find a containing P_{ij}, call it P'_i. (There may be more than one $P_{ij} \ni \boldsymbol{x}$, but any will do.)

Obviously, the intersection $R' = \bigcap_i P'_i$ includes point $\boldsymbol{x}$, but most likely R' has higher dimension than that. Furthermore, since R' is the intersection of exactly one polyhedron per bag it is included in V as well. Hence, we have found a whole polyhedron that includes point $\boldsymbol{x}$.

In the next step, we modify our initial formula f to exclude the polyhedron R', like this:

$$f' = f \wedge \neg R'. \tag{8}$$

Note that we are only adding another assertion to the formula, so we can utilize the incremental mode of SMT solvers to save (a lot of!) time for its next computation.

The important observation here is that we are expanding the original formula f—which describes all solution points—to exclude what we already know to be a solution and continue the search. We can iterate this process until formula f' is unsatisfiable.

This is the algorithm in Python-style pseudocode:

```
# input: a list 'll' of sets of polyhedra.
# output: a list 'rr' of polyhedra.
def compute_prevariety(ll):
    # set the solver to re-use its heuristics
    solver = Solver(incremental=True)
    f = convert_to_SMT_formula(ll)
    rr = []   # results list
    while True:
        # add the formula to the (existing) assertions.
        solver.add_assertion(f)
        # get the model (a variable assignment) that fits the
        # constraints, or None if 'f' is unsatisfiable.
        x = solver.get_model()
        if x is None:
            break
        R = []   # resulting polyhedron
        # cycle through all bags 'B' and
        # collect constraints of polyhedron containing 'x'.
        for B in ll:
            # cycle through all polyhedra 'P' in bag 'B'.
            for P in B:
                if P.contains(x):
                    R.append(P.constraints())
                    break
        # now 'R' defines a polyhedron surely in the intersection.
        # exclude 'R' from further searches.
        f = Not(R)   # new assertion for next round
        rr.append(R)
    return rr  # list of polyhedra.
```

The result of this function is a list of polyhedra. Mathematically, the set of polyhedra describes the equilibrium (resp. prevariety) V. Yet, there are some problems that we address in the next section.

The logic used for SMT formulas is QF_LRA, that is, quantifier-free linear real arithmetic ("real" here means rational). This allows Boolean propositional logic of equations/inequalities consisting of linear polynomials over elements of $\mathbb{Q}$ [BST+10].

3.4. Improvements to the SMT Procedure

Non-maximal Polyhedra

The main issue we experience with the procedure `compute_prevariety()` is that the polyhedron R computed from point $\boldsymbol{x}$ is often not maximal. That is, R is only a lower dimensional face of a higher-dimensional polyhedron. The full high-dimensional polyhedron will eventually be found by the procedure, but earlier-found lower-dimensional faces would still remain in the result list `rr`, albeit superfluous.

To avoid this, we test if each newly found polyhedron R is included in one of the already computed polyhedra of result list `rr`. Unfortunately, this causes quadratic run-time in the number of resulting polyhedra. But there is an observation that can reduce the time needed.

If the newly found polyhedron R is included in some polyhedron R', then obviously, point $\boldsymbol{x} \in R$ is included in R' as well. Testing if a point is included in a polyhedron is simple and fast, so one can test this first and only if this test succeeds one must perform the full polyhedron inclusion test. Measurements show that with this heuristic, almost all polyhedron inclusion tests can be avoided. See Section 5 for details.

In our procedure, we would have to modify function `contains()` in line 22 and function `append()` in line 23 according to these observations.

Redundant Constraints

Another issue is the redundancy of the constraints that are collected in line 23. Efficiency can be increased by minimizing the set of constraints: the larger the number of constraints, the larger the memory demand and, of course, SMT search times.

One can simply cycle through all constraints, test if each of them is really required and if not, drop it. The remaining set is not necessarily a minimal set, though.

Here's how this can be done: Let c be the constraint in consideration, g the formula before and $g' = g \wedge c$ after the addition. If g' is more restrictive than g (i.e., c makes a difference), then the following is unsatisfiable:

$$\begin{aligned} g \wedge \neg g' &= g \wedge \neg(g \wedge c) \\ &= g \wedge (\neg g \vee \neg c) \\ &= (g \wedge \neg g) \vee (g \wedge \neg c) \\ &= g \wedge \neg c. \end{aligned}$$

That formula can be applied by SMT to all constraints to drop superfluous ones.

Preprocessing

Different possibilities were explored to improve the speed of the procedure by preprocessing the input, i.e., the sets of polyhedra.

For one, one can collect all constraints from all bags with only one polyhedron each. Call the resulting polyhedron C. Because of distributivity these constraints hold for all polyhedra of the equilibrium. Hence, we can intersect all polyhedra in their bags with C to test if the intersection is empty, in which case we can drop the polyhedron from its bag and thus reduce problem complexity.

A more potent version of this technique can be used to test polyhedra in all bags on if they are required for the definition of the equilibrium. Let B be the polyhedron in question, A the union of all other polyhedra in B's bag and C the intersection of all other bags. Then the equilibrium is $(A \cup B) \cap C$. If B is required, then $A \cap C \neq (A \cup B) \cap C$ and in particular $A \cap C \subsetneq (A \cup B) \cap C$. Thus, the following set is non-empty:

$$\begin{aligned} ((A \cup B) \cap C) \setminus (A \cap C) &= (A \cup B) \cap C \cap \overline{A \cap C} \\ &= (A \cup B) \cap C \cap (\overline{A} \cup \overline{C}) \\ &= (A \cup B) \cap C \cap \overline{A} \\ &= B \cap C \cap \overline{A}. \end{aligned}$$

This can easily be tested with SMT for each polyhedron B per bag and the superfluous polyhedra can be dropped, again reducing problem complexity.

4. Solving Tropical Equilibrations Over the Integers

4.1. Constraint-based Modeling

The constraint programming (CP) approach is a numerical approach that encodes the tropical equilibration problem as a Constraint Satisfaction Problem (CSP) [SFR14]. This method is implemented in Biocham, which is itself implemented in SWI-Prolog and uses a constraint-solving library over integers.

It doesn't work with a symbolic polyhedral representation of the solutions, but searches for numerical solutions that are points in the intersection of the sets of polyhedra explained earlier. Both complete equilibrations (i.e., for all variables) and partial equilibrations (i.e., for some fast variables) can be computed. In this report and in the following evaluation benchmark, only complete equlibration problems are considered.

Constraint programming, as SMT, deals with combinatorial optimization problems, but it uses constraints in an active manner during search to prune the search space, instead of a symbolic manner to perform clause resolution.

A constraint-based model is defined by a finite set of variables given with their domain, and a finite set of constraints.

- *Variables.* Here the variables are the orders of magnitude $a_i \in \mathbb{Z}$ used to rescale the system by posting $x_i = \bar{x}_i \varepsilon^{a_i}$, as seen in Section 2. Taking integer values ensure that these orders of magnitude are well separated.
- *Constraints.*
  - Tropical equilibration constraints. The degree in ε of the monomials in $\mathcal{P}$ and in $\mathcal{N}$ are integer linear expressions in a_i, the variables of the problem. Such linear expressions are used to filter the domain of the variables by computing their minimum and maximum values according to the bounds of the domain of the variables. Tropical equilibration needs to enforce that the minimum degrees in $\mathcal{P}$ and in $\mathcal{N}$ are equal (1).
  - The constraint of minimum, $A = \min(B, C)$, is implemented with simple inequality constraints $A \leq B$, $A \leq C$, plus reified constraints. A reified constraint is a constraint that links bidirectionnaly the value of one boolean variable X to the satisfaction of another constraint, say on two variables Y and Z, for instance $X \Leftrightarrow Y = Z$ to state that X is true if and only if $Y = Z$. Two reified constraints, $(X \Leftrightarrow A = B) \wedge (Y \Leftrightarrow A = C) \wedge X + Y \leq 1 \wedge A \leq B \wedge A \leq C$, are used to define the minimum constraint to enforce that A is equal to B or C, in addition that A is greater than B and C.
  - Linear invariants. A conservation law is a linear invariant, i.e., an equality beetwen a linear combination of the x_i and a constant. Having the initial values of x_i, and thus the value of the constant, one can add a constraint on the a_i so that the equality of the degrees in the rescaled invariant equation still holds. This allows us to get tropical equilibrations that are compatible with the initial system.

4.2. Constraint-based Search

Constraints come with domain filtering algorithms which dynamically prune the domain of variables when the domain of other variables change in a constraint. However, this strategy is not sufficient to decide the satisfiability of a system of constraints and must be combined with a search procedure for virtually enumerating all possible values of the variables, or more generally, for enumerating all the alternatives of disjunctive constraints.

For solving tropical equilibration problems over finite domains, we obtained good results with dichotomic search by bisecting the domain of the variables (bisect option in SWI-Prolog) without any particular heuristics for choosing the variables. Finite bounds must be given on the values of the variables, but this is not a restriction in practice, since biochemical species' concentrations usually do not vary by more than a hundred of magnitude orders. Furthermore, in order to speed-up the computation of all solutions in such large domains, we used an iterative domain expansion strategy: the problem is first tried with a domain of $[\varepsilon^{-2}, \varepsilon^2]$ for all variables , i.e., equilibrations are searched by rescaling in the $[10^{-2}, 10^2]$ interval. If that fails, the domain is doubled and the problem tried again until the limit interval $[\varepsilon^{-256}, \varepsilon^{256}]$ is reached.

5. Benchmark of Tropical Equilibration Problems from BioModels

This benchmark uses models from the BioModels repository [LNBB$^+$06]. Since BioModels are written in SBML, and to avoid inconsistency in parsing, ODEparse [Lüd20b] was used to convert SBML into ODEs. The resulting ODE systems can be downloaded from ODEbase [LEN$^+$19] at `http://odebase.cs.uni-bonn.de`. A parser to read files from ODEbase was implemented in Biocham. Features of SBML models like events, constraints, time dependency, multiple compartments, or function definitions, were ignored.

The ODE systems selected contain only polynomials or rational functions. For the latter, each equation is multiplied with its principal denominator to get a polynomial. The polynomials are then tropicalized as sketched in Section 2.

The parameter ε was set to $1/11$ for all methods.

Although the three methods can handle partial equilibration problems, we consider here only the problem of complete equilibration, i.e., when all variables equilibrate.

5.1. SMT Method Parameters

The computations work for both equilibria and prevarietyies, since the difference is only in the input. Here, run-times only for computation of equilibria will be shown.

Furthermore, for computation, the logarithms $\log_\varepsilon(k_\alpha)$ in (2) & (3) were rounded to integers or rationals for reasons of numerical stability.

The sets of polyhedra created by tropicalization were saved with polyhedra expressed as sets of equalities and inequalities.

The software used for polyhedral computation was PtCut 3.5.1 [Lüd20c] as available from the web site of the first author. Polyhedral computation were done with the help of the Parma Polyhedra Library (PPL) [BHZ08], version 1.2.

For SMT computation, SMTcut 4.6.0 and the PySMT framework [GM15], version 0.8.0 with SMT engines MathSAT 5.5.1 [CGSS13] and Z3 4.8.4 [DMB08] were used.

Tests were run with Python 3.7.7 on an Intel Core i7-5820K CPU with 48 GB of memory under Windows 10 64-bit. Neither PtCut nor SMTcut make active use of multithreading.

5.2. CP Method Parameters

Concerning the CP solver over the integers, the domain of the degree in epsilon was set to the set $\{0, 1, \ldots, 256\}$. Enlarging this domain had no impact on the results obtained.

In this benchmark we did not include initial values of the variables.

The CP tropicalization was run on Biocham 4.4.14.

5.3. Availability of Code and Data

The used software program, SMTcut, is available under a free software licence from `https://gitlab.com/cxxl/smtcut`. The ODE systems that were used can be downloaded from ODEbase, `http://odebase.cs.uni-bonn.de` and the data that was used as input as well as output resides in one large ($\approx$ 20 MiB) repository available under `https://gitlab.com/cxxl/smtcut_data_1`.

The software Biocham is available at `http://lifeware.inria.fr/biocham4/`, and the data used at `http://lifeware.inria.fr/wiki/Main/Software#DeliverableSYMBIONT`

6. Performance Results

Table 1 shows the run-times for all methods on a subset of models that have complete tropical equilibration solutions over the real numbers, except for model 397 which has no solution.

Table 1: Model number in BioModels, number of variables, and comparison of run-times between PtCut and SMTcut on Windows with MathSAT solver on one side, and constraint programming (CP) in Biocham on the other side. Models marked with a star (*) have a rational vector field. Models with time in italic for CP have no integer tropical equilibration while they have a real-valued equilibration. Timeouts are indicated by a lower bound on the computation time.

BM	R	Dim	Combos	Polyhedra	PtCut [s]	SMTcut [s]	Speed-up	CP Biocham [s]
14		86	10^{67}	> 6996	> 95594.9	> 79964.1	—	14.1
21	*	10	10^{8}	46	3.0	1.4	2.1	> 900
22	*	10	10^{9}	147	2.0	3.7	0.5	> 900
26		11	10^{6}	6	0.1	0.3	0.5	*0.06*
28		16	10^{9}	17	0.4	1.2	0.4	0.2
30		18	10^{10}	6	0.3	0.5	0.8	0.2
32		36	10^{21}	244	110.0	37.2	3.0	*0.6*
41		10	10^{14}	4	1.4	0.3	4.8	> 900
48	*	23	10^{22}	5	29.2	1.4	21.2	12.9

61		22	10^{25}	10084	26408.6	3988.3	6.6	> 900
73		16	10^{14}	13449	> 86400.0	2232.6	> 38.7	> 900
74		19	10^{15}	9685	> 86400.0	2828.7	> 30.5	37.3
93		33	10^{26}	596	6113.4	169.4	36.1	1.2
102		13	10^{10}	322	7.2	23.4	0.3	0.3
103		17	10^{18}	1938	763.5	397.7	1.9	0.6
105		27	10^{23}	130	644.2	18.1	35.6	*1.5*
147		24	10^{16}	54	18.7	8.3	2.3	14.4
151		66	10^{44}	> 17784	> 86400.0	> 75494.4	—	*3.1*
183		67	10^{84}	1	> 85072.9	94.9	> 896.2	> 900
200		22	10^{20}	20	64.0	4.1	15.5	> 900 (190.3)
221	*	8	10^{9}	50	2.1	1.3	1.6	57.3
222	*	8	10^{9}	192	11.0	6.0	1.8	> 900
230		24	10^{16}	68	24.6	11.2	2.2	*0.7*
315		19	10^{16}	13	2.5	1.9	1.3	0.4
328	*	18	10^{9}	86	1.4	4.1	0.3	7.3
365		30	10^{23}	70	583.7	15.6	37.5	0.5
396	*	36	10^{28}	54	21.7	9.8	2.2	> 900
397	*	50	10^{38}	0	14.9	0.2	79.4	*1.2*
407		47	10^{16}	212	968.2	35.2	27.5	*2.0*
410		53	10^{41}	> 28115	> 86400.0	> 78622.5	—	> 900
430		27	10^{13}	1676	58.0	282.5	0.2	0.4
431		27	10^{16}	155	13.2	18.4	0.7	0.4
477		43	10^{17}	467	571.0	77.7	7.3	*5.8*
482	*	23	10^{11}	17	2.3	1.1	2.2	31.6
489		35	10^{21}	42	57.5	9.3	6.2	> 900
491		57	10^{24}	1	17.2	0.4	39.4	6.3
492		52	10^{25}	1	2.9	0.5	6.3	6.2
498	*	19	10^{10}	214	3.3	12.4	0.3	3.3
501		35	10^{25}	916	> 89107.6	237.7	> 374.8	*0.6*
560		59	10^{29}	> 21712	> 86400.0	> 80237.8	—	*2.3*
576		34	10^{18}	756	55.8	96.2	0.6	3.3
599		30	10^{16}	24	5.8	4.7	1.2	*0.5*
637		12	10^{12}	12	4.2	1.2	3.5	*0.3*
638		21	10^{27}	13	32.3	4.9	6.6	*1.1*
666	*	34	10^{24}	64	5.6	4.3	1.3	> 900
730	*	45	10^{47}	> 31349	> 86400.0	> 84098.0	—	> 900

The run-times value our attention for certain models:

- BioModels 14, 151, 410, 560 & 730: each of them could not be computed with PtCut and it is likely infeasible to that with this approach. SMTcut was at least able to compute many polyhedra, even though it is unknown how large a part of the full equilibrium this constitutes. On the other hand, the CP method found integer solutions for model 14, and proved that models 151 and 560 are unsatisfiable over the integers in less than 3 seconds.
- BioModels 183 and 491, 492: Here SMTcut was able to play out its full potential: with only one polyhedron in the equilibrium, it took only some rounds until the maximal polyhedron was found. PtCut, on the other hand, terminated after a day's work with an intermediate number of 15000 polyhedra and still only 5 of 65 iterations done. The CP method failed to solve model 183 in less than 15 minutes, but found integer solutions for the other two models in 6 seconds.
- BioModel 397 doesn't have a solution at all. SMTcut realizes this in 0.2 s, whereas CP takes 1.2 s and PtCut almost 15 s.
- BioModels 48, 73, 74, 93, 105, 407, and especially 501: computation time of SMTcut was at least 20 times lower than with PtCut. The CP method was even faster by one or two orders of magnitude on models 74, 93, 105 and 407, but was in timeout on model 73.

- BioModels 102, 328, 430 & 498: here PtCut was at least 3 times faster than SMTcut. The reason might be that for models of medium dimension (< 20) and with many polyhedra, PtCut can be faster if the intermediate expression swell is not too large, see Table 4. All these models were solved by CP in less than a second or 7.3 s

Over the reals, SMTcut was the better choice in all cases where PtCut needed more than 58 s of computation time. Over the integers, the CP method could be even faster to conclude by up to two orders of magnitude, but could also fail with a timeout of 2 minutes whereas SMTcut took a few seconds. The CP method reveals that nearly 30% of the tropical equilibrations satisfiable over $\mathbb{R}$ have no solution over $\mathbb{Z}$.

6.1. Further Results with the CP Method

The complete equilibrations were computed and the existence of a tropical equilibrium or not was reported in the Tables 1, 2 and 3.

Results for all the polynomial models are presented in Table 2. On the over 150 models, only one had a timeout over 15 minutes of computation, less than 10% ran over 2 seconds, with one of them running over 20 seconds. The extreme majority of the models took then less than 2 seconds to be computed.

Rational functions models results, presented in Table 3, were way slower than original polynomial models (multiplication time not taken in account in the table) to tropicalize. The multiplication by the common denominator, even after simplification, leaves large equations to work with, and was involved in the slow-down of the computation. A timeout over 15 minutes happens for more than half the models, and often exceed more than an hour of computation. The computation is taking more than 20 seconds to compute in around one eight of the cases not timed-out.

6.2. Further Results for the SMT Method

6.2.1. Benchmarking of Different SMT Solvers

Under Windows, we could only use MathSAT [CGSS13] and Z3 [DMB08] and were unable to install the CVC4 [BCD+11] and Yices [Dut14] solvers. The other solvers that are in principle supported by PySMT are CUDD, PicoSAT and Boolector, but they don't support the QF_LRA logic, so they were no option.

Hence, we switched to Ubuntu Linux 18.04 64-bit and benchmarked some models again on the same machine to get a comparison. Under Ubuntu we could work with MathSAT 5.5.1, Z3 4.8.4 and Yices 2.6.0. Unfortunately, CVC4 1.7-prerelease could be installed, but didn't work properly.

Table 5 shows the run-times for selected models.

A side-by-side comparison of operating systems suggests that Windows and Ubuntu implementations are similar. MathSAT on Windows was some 3–9% slower than on Ubuntu, whereas Z3 was not always faster on any one operating system.

A comparison of MathSAT and Z3 solvers suggests that they yield comparable run-times, with Z3 in one case almost twice as fast as MathSAT (BioModel 73). Yices could not be evaluated properly, since it crashed with all larger models.

6.2.2. Preprocessing

Table 6 shows a comparison of times with and without preprocessing the input. Several observations can be made:

- BioModel 183: the process was killed after over 11 hours of preprocessing. This is due to the extraordinary number of polyhedra in some input bags.
- With the exception of BioModel 74 (and, of course, 183), all models with a run-time over 24 s ran faster with preprocessing (including the time for preprocessing itself) than without.
- All models with run-times less than 17 s ran slower (or as fast in the case of BioModels 22 & 498) with preprocessing than without.

So it seems that this kind of preprocessing is advisable only in special cases.

6.2.3. Profiling

The relative time needed in different parts of the procedure to compute the whole equilibrium varies with the number of resulting polyhedra. Table 7 shows the relative time spent in three different parts of the procedure:

- `S`: Searching for another point outside the already known polyhedra,
- `M`: Minimizing the found polyhedron,
- `I`: Inserting that polyhedron into the list of already known ones and testing for inclusion.

From the numbers it is obvious that the inclusion check time takes relatively more time as the number of polyhedra grows. Since that routine requires quadratic time, this is as expected.

Furthermore, the SMT checking for every next point needs more time as the number of polyhedra grows as well. This is due to the fact that the formula grows over time (to exclude all already found polyhedra) and thus the search gets more complicated, and we should not forget that we would expect an exponential run-time from theory.

The time needed for minimization, in contrast, is getting less prominent, which is no surprise, since the time required to check for superfluous constraints in a polyhedron does not depend on the number of already found polyhedra, nor do we expect to find polyhedra with more constraints as the search takes more time.

6.2.4. Polyhedral Inclusion Testing

As was described in Section 3.3, the test for inclusion of a newly found polyhedron in the set of already found ones can be sped up by testing containment of the included point $\boldsymbol{x}$, which was found by the SMT solver.

Some cursory investigation shows that the number of full checks that still have to be done is about 0.26–0.68 times the number of polyhedra in the result list.

Yet, the main message is that even though the constant is low, the run-time of the inclusion test is still quadratic and it becomes the most dominant part of the computation for large result lists.

7. Conclusion

We have shown that tropical equilibration problems of real size, coming from the curated part of the BioModels model repository, can be solved quite efficiently both over the integers and over the real numbers.

For solving over the reals, a new method, based on SMT, was presented and compared to a known algorithm using purely polyhedral methods. The run-times were always smaller for problems that would otherwise take more than 58 seconds to compute, sometimes by a factor of 25 or more. The novel method has the further advantage that, even if computation of the entirety of the solution is infeasible, parts of it can be computed and more computation power would lead to more parts being computed. Furthermore, extensive benchmarks with different SMT solvers under different operating systems were run on tropicalizations of 46 different BioModels. The choice of operating systems doesn't make a significant difference. The choice of SMT solver sometimes makes a difference, but there is no clear winner there. Only MathSAT and Z3 worked under all circumstances and had comparable run-times.

Tropical solutions over the integers ensure that the orders of magnitude of the variables are well separated. We have shown that the Constraint Programming method introduced in [SFR14] exhibits good performances similarly to the SMT method on this benchmark, with run-times mostly inferior to 2 minutes per model. However, this evaluation in BioModels revealed that 30% of the tropical equilibration problems solvable over the real numbers have no integer solution.

Both the SMT and CP methods thus appear to be scalable on BioModels. More work is now needed to compare the real solutions with the integer solutions found by the two methods, and assess the scope of their respective use for model reduction applications.

References

[BCD+11] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovi'c, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV '11)*, volume 6806 of *Lecture Notes in Computer Science*, pages 171–177. Springer, July 2011. Snowbird, Utah.

[BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. *Sci. Comput. Program.*, 72(1-2):3–21, June 2008.

[BST+10] Clark Barrett, Aaron Stump, Cesare Tinelli, et al. The smt-lib standard: Version 2.0. In *Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, England)*, volume 13, page 14, 2010.

[CGSS13] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In Nir Piterman and Scott Smolka, editors, *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer, 2013.

[DMB08] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[DMB11] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

[Dut14] Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer-Aided Verification (CAV'2014)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, July 2014.

[GM15] Marco Gario and Andrea Micheli. PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms. In *SMT Workshop 2015*, 2015.

[LEN+19] Christoph Lüders, Hassan Errami, Matthias Neidhardt, Satya S. Samal, and Andreas Weber. ODEbase: an extensible database providing algebraic properties of dynamical systems. `http://wrogn.com/wp-content/uploads/lueders-casc-2019-odebase.pdf`, 2019.

[LNBB+06] Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L. Snoep, and Michael Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(Database issue):D689–D691, Jan 2006.

[Lüd20a] Christoph Lüders. Computing Tropical Prevarieties with Satisfiability Modulo Theories (SMT) Solvers. *arXiv preprint arXiv:2004.07058*, April 2020.

[Lüd20b] Christoph Lüders. ODEparse: generate ODEs from SBML. `https://gitlab.com/cxxl/odeparse/`, 2020.

[Lüd20c] Christoph Lüders. PtCut: Calculate Tropical Equilibrations and Prevarieties. `http://www.wrogn.com/ptcut/`, 2020.

[Mon16] David Monniaux. A survey of satisfiability modulo theory. In *International Workshop on Computer Algebra in Scientific Computing*, pages 401–425. Springer, 2016.

[SFR14] Sylvain Soliman, François Fages, and Ovidiu Radulescu. A constraint solving approach to model reduction by tropical equilibration. *Algorithms for Molecular Biology*, 9(24), December 2014.

[SGF+15] Satya Swarup Samal, Dima Grigoriev, Holger Fröhlich, Andreas Weber, and Ovidiu Radulescu. A geometric method for model reduction of biochemical networks with polynomial rate functions. *Bulletin of Mathematical Biology*, 77(12):2180–2211, October 2015.

[Zie95] Günter M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1995.

A. Run-times

Table 2: Satisfiability over the integers and run-times in seconds with CP in Biocham for the *polynomial* models of the curated part of BioModels.

BM	output	Biocham [t]
2	sat	0.2
5	sat	0.1
9	sat	0.3
11	unsat	0.3
14	sat	14.1
26	unsat	0.1
28	sat	0.2
30	sat	0.2
35	unsat	0.1
38	sat	0.1
40	sat	0.01
50	unsat	0.1
52	unsat	0.04
57	unsat	0.1
60	sat	0.01
69	unsat	0.1
72	sat	0.03
80	sat	0.1
82	sat	0.05
85	sat	0.3
86	sat	0.5
91	unsat	0.2
92	unsat	0.01
99	unsat	0.1
102	sat	0.3
103	sat	0.6
105	unsat	1.5
108	sat	0.2
123	sat	0.3
147	sat	14.4
150	sat	0.01
156	sat	0.01
159	sat	0.01
163	unsat	0.2
173	unsat	0.4
178	unsat	0.02
197	unsat	0.1
198	sat	0.05
199	sat	0.05
200	sat	190.3
205	unsat	18.0
220	unsat	1.3
226	unsat	0.3
229	unsat	0.1
230	unsat	0.7
233	unsat	0.01
243	unsat	0.1
257	sat	0.1
259	unsat	0.2
260	unsat	0.2
261	unsat	0.2
262	unsat	0.1
263	unsat	0.1
264	sat	0.1
267	unsat	0.01
270	sat	0.4
271	unsat	0.04
272	unsat	0.04
282	unsat	0.01
283	unsat	0.01
286	unsat	2.2
289	unsat	0.04
292	unsat	0.01
306	unsat	0.02
307	unsat	0.01
309	sat	0.01
310	unsat	0.01
311	unsat	0.01
312	unsat	0.01
314	unsat	0.05
315	sat	0.4
321	unsat	0.02
332	unsat	2.7
333	unsat	1.2
334	unsat	2.4
335	unsat	0.4
357	unsat	0.04
359	unsat	0.1
360	unsat	0.05
361	sat	0.04
362	unsat	0.5
363	unsat	0.01
364	unsat	0.1
365	sat	0.5
389	unsat	0.2
405	sat	0.1
407	unsat	2.0
413	unsat	0.02
416	unsat	0.7
430	sat	0.4
431	sat	0.4
459	unsat	0.01
460	unsat	0.01
475	unsat	0.2
478	unsat	0.5
483	sat	0.1
484	unsat	0.002
485	sat	0.01
486	unsat	0.003
487	unsat	0.02
491	unsat	6.3
492	unsat	6.2
500	unsat	0.1
501	unsat	0.6
504	unsat	4.9
519	unsat	0.1
530	sat	0.1
531	sat	0.01
532	sat	0.005
546	unsat	0.02
552	unsat	0.01
553	unsat	0.01
555	sat	0.001
559	unsat	3.3
561	sat	0.002
566	sat	0.002
567	sat	0.002
569	sat	5.6
580	sat	0.4
581	unsat	0.3
582	unsat	0.6
584	unsat	0.2
599	unsat	0.5
609	unsat	0.02
614	sat	0.01
619	unsat	0.2
629	sat	0.01
630	sat	0.03
635	sat	4.8
636	timeout	> 900
637	unsat	0.3
638	unsat	1.1
640	unsat	0.8
646	sat	19.6
647	sat	0.1
651	sat	0.3
658	unsat	0.2
661	sat	0.1
663	sat	0.03
667	unsat	5.0
676	unsat	0.1
678	sat	0.01
687	unsat	0.1
688	unsat	0.1
692	unsat	0.04
705	sat	1.1
707	sat	0.02
708	sat	0.03
709	sat	0.1
710	sat	0.05
716	sat	0.02
721	sat	0.1
726	sat	0.1

Table 3: Satisfiability over the integers and run-times in seconds with CP in Biocham for the *rational fractional* models of the curated part of BioModels.

BM	output	Biocham [t]
10	timeout	> 900
16	sat	175.4
17	timeout	> 900
21	timeout	> 900
22	timeout	> 900
23	timeout	> 900
27	sat	0.2
29	sat	1.1
31	sat	482.4
32	unsat	0.6
33	unsat	1.1
37	unsat	0.1
39	sat	0.5
41	timeout	> 900
42	timeout	> 900
43	sat	0.4
44	timeout	> 900
45	unsat	6.2
46	timeout	> 900
48	sat	12.9
49	unsat	42.7
67	sat	111.4
73	timeout	> 900
74	sat	37.3
79	sat	0.1
84	timeout	> 900
90	unsat	0.8
93	sat	1.2
98	unsat	2.6
107	timeout	> 900
110	sat	0.5
113	sat	170.4
114	sat	0.04
115	sat	0.05
116	sat	133.7
145	timeout	> 900
151	unsat	3.1
157	sat	0.01
166	sat	3.8
167	unsat	0.2
168	sat	0.7
169	sat	0.5
170	sat	0.1
181	sat	2.4
182	unsat	17.2
191	sat	39.3
192	sat	0.3
201	unsat	2.8
203	sat	13.8
204	sat	14.0
206	sat	1.7
209	sat	7.5
210	sat	2.8
215	sat	51.4
216	timeout	> 900
221	sat	57.3
224	sat	0.3
228	timeout	> 900
231	unsat	0.1
240	sat	2.2
242	sat	1.7
249	sat	2.9
258	sat	0.1
269	unsat	0.5
275	sat	0.1
284	timeout	> 900
290	sat	0.1
294	sat	10.6
296	sat	6.0
300	sat	2.1
308	unsat	0.1
313	unsat	0.1
319	unsat	851.1
320	unsat	0.2
323	sat	0.02
325	sat	21.3
328	sat	7.3
329	sat	0.1
347	unsat	2.6
351	sat	0.8
352	sat	0.8
358	unsat	0.1
366	unsat	0.1
388	timeout	> 900
394	unsat	1.0
395	unsat	0.4
397	unsat	1.2
409	unsat	2.8
414	sat	0.01
415	unsat	0.6
423	sat	0.1
424	unsat	5.2
425	sat	0.03
427	unsat	1.0
434	unsat	1.8
435	sat	0.4
438	unsat	0.1
447	timeout	> 900
452	unsat	6.4
453	unsat	8.9
454	sat	2.4
455	sat	1.1
456	sat	2.6
458	sat	0.1
461	sat	3.4
462	unsat	0.1
467	unsat	0.7
477	unsat	5.8
482	sat	31.6
489	timeout	> 900
495	sat	38.5
498	sat	3.3
517	sat	0.6
518	sat	0.6
523	unsat	0.6
524	unsat	0.6
525	unsat	1.0
526	unsat	1.0
539	timeout	> 900
543	unsat	11.8
544	unsat	10.4
545	sat	0.4
557	sat	0.9
560	unsat	2.3
573	sat	0.02
576	sat	3.3
583	unsat	0.7
611	unsat	1.7
622	sat	0.2
623	unsat	1.2
624	unsat	0.1
625	timeout	> 900
626	sat	0.2
631	unsat	0.4
639	sat	0.1
689	sat	0.1
690	sat	0.1
704	unsat	0.2
713	sat	0.3
717	unsat	0.2
720	timeout	> 900
724	unsat	1.5
728	sat	0.01
729	sat	0.1
732	sat	0.3
737	sat	90.3
738	sat	84.8

Table 4: Intermediate expression swell for some BioModels computed with the SMT method.

BM	Combos	PtCut [s]	Polyhedra	Max. int. ph's
21	10^{8}	2.978	46	3408
22	10^{9}	2.036	147	1170
32	10^{21}	109.952	244	1092
41	10^{14}	1.416	4	924
48	10^{22}	29.175	5	1160
93	10^{26}	6113.411	596	47772
102	10^{10}	7.199	322	4784
103	10^{18}	763.480	1938	111402
105	10^{23}	644.176	130	21088
147	10^{16}	18.738	54	5069
200	10^{20}	63.969	20	4704
221	10^{9}	2.150	50	2573
222	10^{9}	10.968	192	12516
230	10^{16}	24.554	68	3330
315	10^{16}	2.529	13	432
328	10^{9}	1.351	86	140

365	10^{23}	583.655	70	15030
396	10^{28}	21.668	54	972
407	10^{16}	968.175	212	15010
430	10^{13}	58.050	1676	4683
431	10^{16}	13.193	155	984
477	10^{17}	570.995	467	23460
482	10^{11}	2.342	17	495
489	10^{21}	57.451	42	4824
498	10^{10}	3.331	214	1750
576	10^{19}	55.783	756	10752
599	10^{16}	5.796	24	456
637	10^{12}	4.208	12	1140
638	10^{27}	32.306	13	2124
666	10^{23}	5.633	64	464

Table 5: Run-times of SMT method for some models under Ubuntu (U) compared to Windows (W). A star (*) signifies a program crash.

BioModel	Polyhedra	MathSAT (U) [s]	Z3 (U) [s]	Yices (U) [s]	MathSAT (W)	Z3 (W) [s]
22	147	3.583	4.656	2.097	3.703	4.000
32	244	34.598	42.938	35.319	37.234	37.156
73	13449	2089.749	1314.049	*	2232.641	1288.156
93	596	154.564	198.951	*	169.375	199.156
183	1	90.184	84.033	131.141	94.922	102.938
397	0	0.179	0.160	0.132	0.188	0.156
430	1676	260.927	248.023	*	282.484	260.625
501	916	216.917	297.374	*	237.719	292.672

Table 6: Run-times of SMT method in seconds for some models without and with preprocessing using the MathSAT solver under Windows. Column "PP time" contains the time for preprocessing and column "after PP" the total time minus the preprocessing time.

BioModel	Dim	Combos	Polyhedra	Without PP	With PP	PP time	After PP	Speed-up
21	10	10^8	46	1.3	1.5	0.6	1.0	0.87
22	10	10^9	147	3.7	3.7	0.5	3.3	1.00
26	11	10^6	6	0.2	0.4	0.3	0.1	0.50
28	16	10^9	17	1.2	1.8	0.8	1.0	0.67
30	18	10^{10}	6	0.4	1.3	1.0	0.3	0.31
32	36	10^{21}	244	37.2	29.8	1.8	28.0	1.25
41	10	10^{14}	4	0.3	2.0	1.9	0.1	0.15
48	23	10^{22}	5	1.2	12.5	12.3	0.2	0.10
61	22	10^{25}	10084	3191.3	2651.9	40.6	2611.3	1.20
73	16	10^{14}	13449	2232.6	2203.2	1.4	2201.8	1.01
74	19	10^{15}	9685	1873.7	2179.1	1.2	2177.8	0.86
93	33	10^{26}	596	169.4	143.8	7.3	136.5	1.18
102	13	10^{10}	322	24.6	20.9	2.4	18.5	1.18
103	17	10^{18}	1938	382.8	343.1	38.2	304.8	1.12
105	27	10^{23}	130	16.6	73.8	60.2	13.6	0.22
147	24	10^{16}	54	11.7	12.3	6.0	6.3	0.95
183	67	10^{84}	1	94.9	> 40000.0	> 40000.0	—	—
200	22	10^{20}	20	3.9	10.1	8.0	2.2	0.39
221	8	10^9	50	1.1	2.1	1.2	0.9	0.52
222	8	10^9	192	6.1	6.9	1.5	5.4	0.88
230	24	10^{16}	68	9.7	17.0	7.1	9.9	0.57
315	19	10^{16}	13	1.6	4.0	2.9	1.1	0.40
328	18	10^9	86	3.8	4.0	0.6	3.4	0.95
365	30	10^{23}	70	14.2	18.1	10.6	7.5	0.78
397	50	10^{38}	0	0.2	1.8	1.8	0.0	0.11
407	47	10^{16}	212	37.1	30.2	2.0	28.1	1.23
430	27	10^{13}	1676	282.5	255.8	1.6	254.2	1.10
431	27	10^{16}	155	17.7	18.3	3.2	15.1	0.97
477	43	10^{17}	467	59.0	55.0	1.7	53.3	1.07
482	23	10^{11}	17	1.0	2.0	1.2	0.8	0.50
489	35	10^{21}	42	7.5	22.3	14.8	7.5	0.34

491	57	10^{24}	1	0.4	3.0	3.0	0.0	0.13
492	52	10^{25}	1	0.4	3.0	3.0	0.0	0.13
498	19	10^{10}	214	11.3	11.3	1.9	9.4	1.00
501	35	10^{25}	916	237.7	183.0	7.4	175.5	1.30
576	34	10^{18}	756	95.2	91.9	4.0	87.9	1.04
599	30	10^{16}	24	4.1	5.8	3.6	2.3	0.71
637	12	10^{12}	12	1.1	2.3	1.4	0.9	0.48
638	21	10^{27}	13	4.5	41.8	39.9	2.0	0.11
666	34	10^{24}	64	4.2	5.5	1.4	4.1	0.76

Table 7: Relative run-time of SMT method used for `S`earching another point, `M`inimizing a polyhedron and `I`nclusion checking. The second part of the models didn't finish and the numbers signify only the times until the process was terminated.

BioModel	Polyhedra	% in `S`	% in `M`	% in `I`
26	6	15.3	46.3	0.0
22	147	15.6	59.5	7.2
222	192	20.1	53.7	11.2
498	214	18.5	62.9	9.4
407	212	9.8	74.3	11.1
32	244	10.0	72.0	12.6
477	467	12.2	65.0	16.2
93	596	13.5	72.0	10.5
501	916	12.9	76.4	7.5
430	1676	23.3	53.8	18.5
103	1938	27.5	56.2	10.9
74	9685	42.5	15.7	32.7
61	10084	53.4	18.6	21.6
73	13449	43.3	13.1	34.4
14	6996	16.0	55.0	26.1
151	17784	28.3	22.2	45.1
560	21712	18.1	11.1	68.1
410	28115	34.4	24.8	35.0
730	31349	39.0	11.3	44.9