PtCut:
A Program To Compute Tropical Prevarieties

Christoph Lüders

Universität Bonn

SYMBIONT meeting, 22 March 2018, Bonn

Mission Statement

- Input: a polynomial system of equations.
- (This system is derived from ODEs, which are derived from a (bio)chemical reaction network with mass-action kinetics.)
- Rate constants are symbolic, but values have to be specified for computation.
- Output: the tropical prevariety of the system.
- We want to deduce approximate knowledge about the classical variety from the study of its tropical prevariety.

Tropicalization (1)

- First, the system is tropicalized. This means, each '$\cdot$' becomes a '$+$' and each '$+$' becomes a 'min'. There is no translation for '$-$'!
- The multiplicative constant (i.e., the replaced rate constants) is transformed through a $\log_\varepsilon(x)$, $\varepsilon < 1$, and rounded to integer.
- Example: the classical equation $x_1 + 3x_2 + x_3^5$ becomes $\min(\hat{x}_1, \log_\varepsilon 3 + \hat{x}_2, 5\hat{x}_3)$ in the tropical world.
- (By abuse of language, I'll often keep the variable names: $\mathrm{Trop}(x_4 + x_5^4) = \min(x_4, 4x_5)$.)
- Choice of ε and rounding precision has influence on the number and shape of solutions and run-time.

Tropicalization (2)

- A classical root is found if $f(x) = 0$. That is, positive and negative monomials cancel each other out: $\sum_i P_i = \sum_j N_j,\ P_i, N_j \geq 0$.
- Since there is no tropical subtraction, a *tropical root* is defined as the points where the minimum of tropical monomials is attained at least twice:

$$P_k \geq P_i = N_j \leq N_\ell. \tag{1}$$

- Cycling over all pairs of monomials and applying (1) defines a *tropical hypersurface*, which in turn defines a set of polyhedra.
- The intersection of the tropical hypersurfaces of multiple polynomials f_i is called a *tropical prevariety*.
- For "complex" solutions, we ignore the sign of monomials:

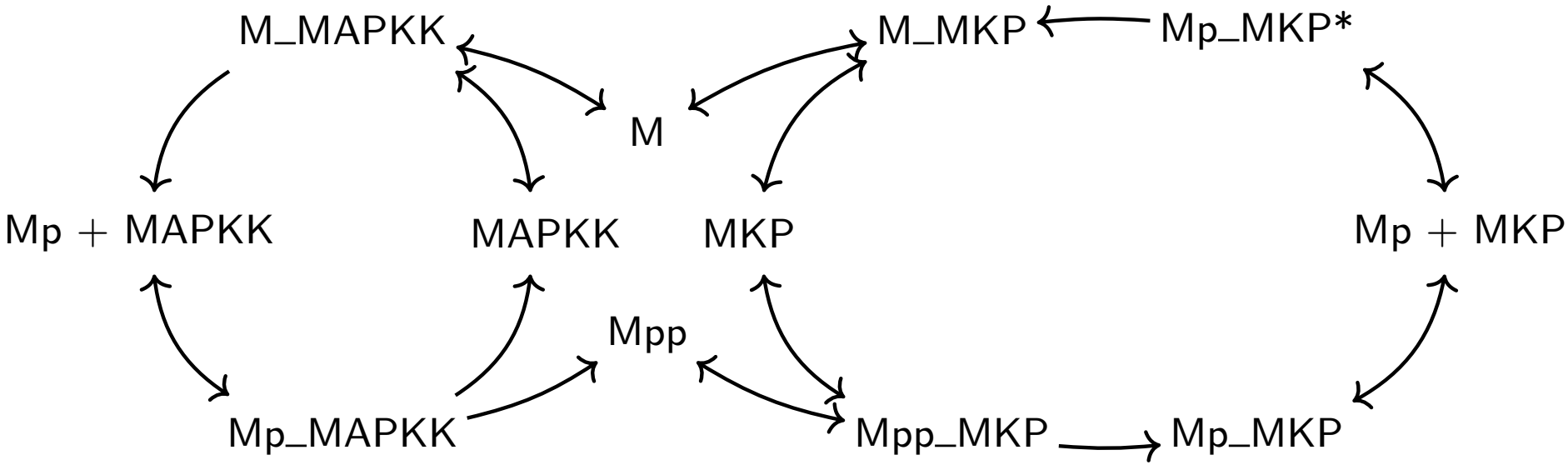
$$P_i = P_j \leq P_k,\ \ i \neq j.$$

Program Overview

- Polynomial systems often come from the Biomodels database (in which case a SBML-parser is needed to build the ODEs from SBML.) The database specifies rate constants as well.
- PtCut reads this polynomial system as a list of polynomials f_r in x_i with parameters k_j. PtCut tries to solve the system of equations $f_r = 0$.
- Parameters are replaced, then polynomials are tropicalized and thus create sets of polyhedra.
- These sets of polyhedra are intersected one by one to get the tropical prevariety.
- Some tricks help to keep the run-time low (we'll come to that).

Example: Biomodel 26 (1)

This is the MAPK model (Markevich et al., 2004) as modeled in the BioModels database.



Example: Biomodel 26 (2)

From the above network one can derive the following ODEs:

$$
\begin{aligned}
\dot{x}_1 &= k_2 x_6 + k_{15} x_{11} - k_1 x_1 x_4 - k_{16} x_1 x_5 \\
\dot{x}_2 &= k_3 x_6 + k_5 x_7 + k_{10} x_9 + k_{13} x_{10} - x_2 x_5 (k_{11} + k_{12}) - k_4 x_2 x_4 \\
\dot{x}_3 &= k_6 x_7 + k_8 x_8 - k_7 x_3 x_5 \\
\dot{x}_4 &= x_6 (k_2 + k_3) + x_7 (k_5 + k_6) - k_1 x_1 x_4 - k_4 x_2 x_4 \\
\dot{x}_5 &= k_8 x_8 + k_{10} x_9 + k_{13} x_{10} + k_{15} x_{11} - x_2 x_5 (k_{11} + k_{12}) - k_7 x_3 x_5 - k_{16} x_1 x_5 \\
\dot{x}_6 &= k_1 x_1 x_4 - x_6 (k_2 + k_3) \\
\dot{x}_7 &= k_4 x_2 x_4 - x_7 (k_5 + k_6) \\
\dot{x}_8 &= k_7 x_3 x_5 - x_8 (k_8 + k_9) \\
\dot{x}_9 &= k_9 x_8 - k_{10} x_9 + k_{11} x_2 x_5 \\
\dot{x}_{10} &= k_{12} x_2 x_5 - x_{10} (k_{13} + k_{14}) \\
\dot{x}_{11} &= k_{14} x_{10} - k_{15} x_{11} + k_{16} x_1 x_5
\end{aligned}
$$

Some additional calculations yield the following conservation laws:

$$
\begin{aligned}
k_{17} &= x_5 + x_8 + x_9 + x_{10} + x_{11} \\
k_{18} &= x_4 + x_6 + x_7 \\
k_{19} &= x_1 + x_2 + x_3 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}
\end{aligned}
$$

Example: Biomodel 26 (3)

Setting all differentials to zero, we only keep the polynomials as input for PtCut. The parameters are just extracted from BioModels database.

`Polynomial_system.txt`:

```
k2*x6 + k15*x11 - k1*x1*x4 - k16*x1*x5
k3*x6 + k5*x7 + k10*x9 + k13*x10 - x2*x5*(k11 + k12) - k4*x2*x4
k6*x7 + k8*x8 - k7*x3*x5
x6*(k2 + k3) + x7*(k5 + k6) - k1*x1*x4 - k4*x2*x4
k8*x8 + k10*x9 + k13*x10 + k15*x11 - x2*x5*(k11 + k12) - k7*x3*x5 - k16*x1*x5
k1*x1*x4 - x6*(k2 + k3)
k4*x2*x4 - x7*(k5 + k6)
k7*x3*x5 - x8*(k8 + k9)
k9*x8 - k10*x9 + k11*x2*x5
k12*x2*x5 - x10*(k13 + k14)
k14*x10 - k15*x11 + k16*x1*x5
x5 - k17 + x8 + x9 + x10 + x11
x4 - k18 + x6 + x7
x1 - k19 + x2 + x3 + x6 + x7 + x8 + x9 + x10 + x11
```

`Params.txt`:

```
k1 = 0.02
k2 = 1
k3 = 0.01
k4 = 0.032
k5 = 1
k6 = 15
k7 = 0.045
k8 = 1
k9 = 0.092
k10 = 1
k11 = 0.01
k12 = 0.01
k13 = 1
k14 = 0.5
k15 = 0.086
k16 = 0.0011
k17 = 100
k18 = 50
k19 = 500
```

Combining All Polytopes (1)

- Each tropical polynomial gives rise to a tropical hypersurface that describes a set (a disjunction) of polyhedra:

$$B_i = \bigcup_j P_{ij}, \quad i \in [1 : n].$$

- The intersection (conjunction) of all tropical hypersurfaces is the tropical prevariety U:

$$U = \bigcap_i^n B_i = \bigcap_i^n \bigcup_j P_{ij}.$$

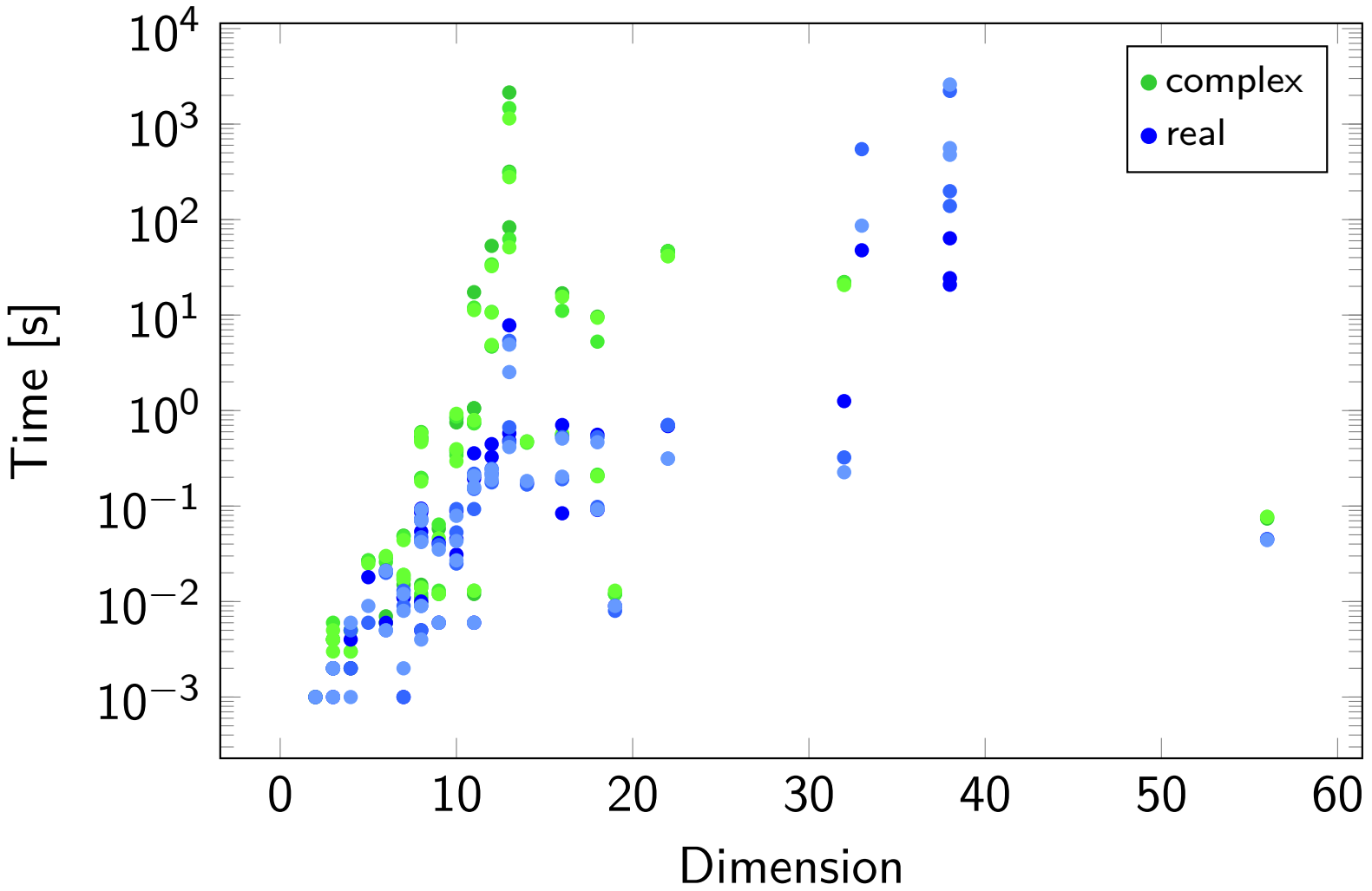
- Because of distributivity, we can rewrite that as:

$$U = \bigcup_{\ell \in \mathcal{I}} \bigcap_i^n P_{i\ell_i}, \quad \text{with } \mathcal{I} = \bigtimes_i^n |B_i|.$$

Combining All Polytopes (2)

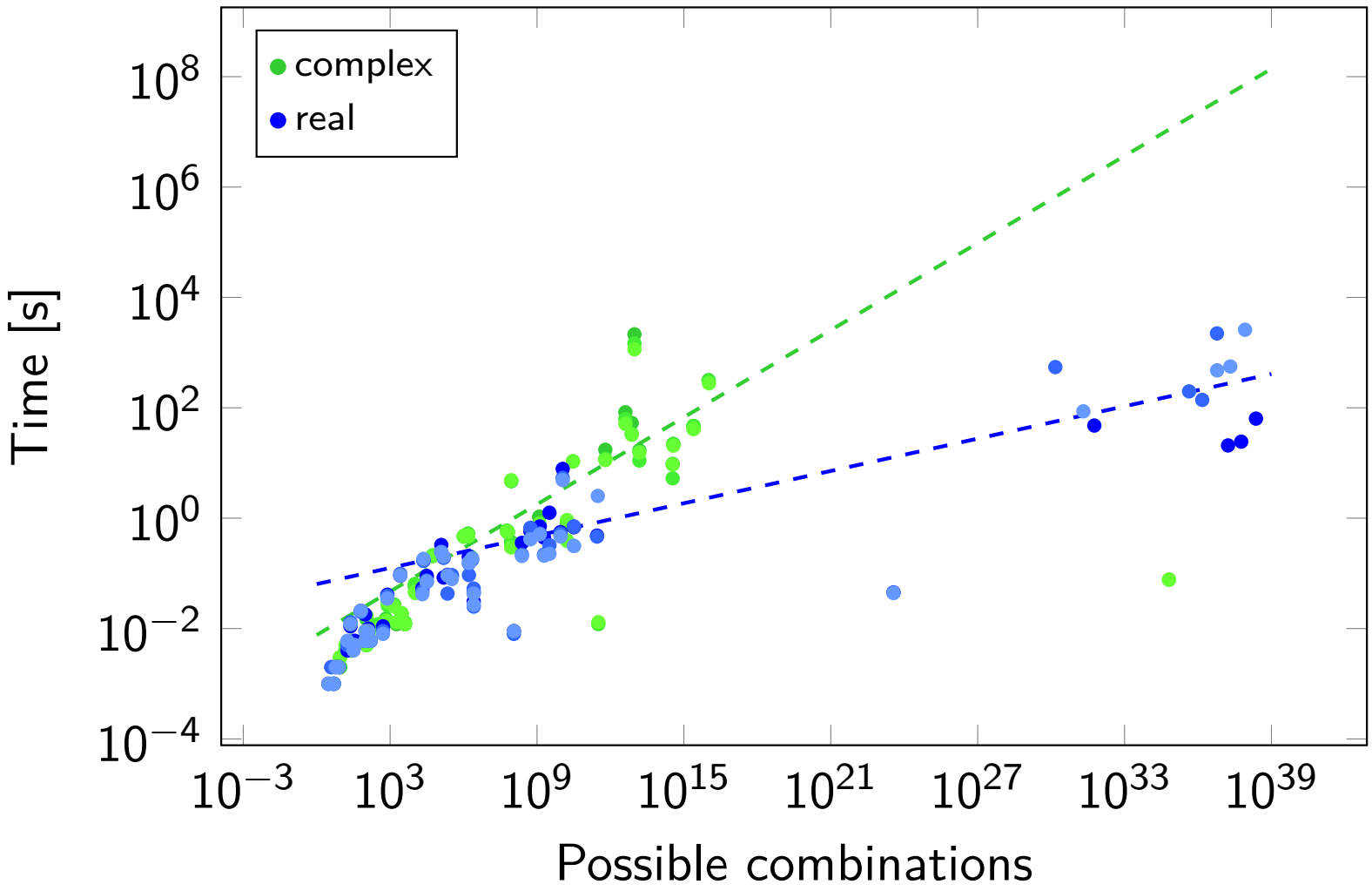
- The number of *possible* combinations is exponential in n.
- However, the biological models we examined so far have usually a very small (< 10) number of polyhedra in the prevariety, with only one exception (Biomodel 102), where it is ≈ 400.
- Still, the number of *intermediate* polyhedra can be much higher. Keeping this number low is important for fast computations. This can be done by several methods:
  - If, after intersecting two hypersurfaces, for two of the resultant polyhedra $P \subseteq Q$ holds, we keep only Q.
  - The order of evaluation is important! A good heuristic is to intersect hypersurfaces with the fewest polyhedra first.
  - If we find constraints that are shared by all polyhedra of a hypersurface, they must be shared by the polyhedra of the prevariety as well. Hence, we can apply those constraints to all unprocessed hypersurfaces to reduce the number of polyhedra in them.

Time vs. Dimension



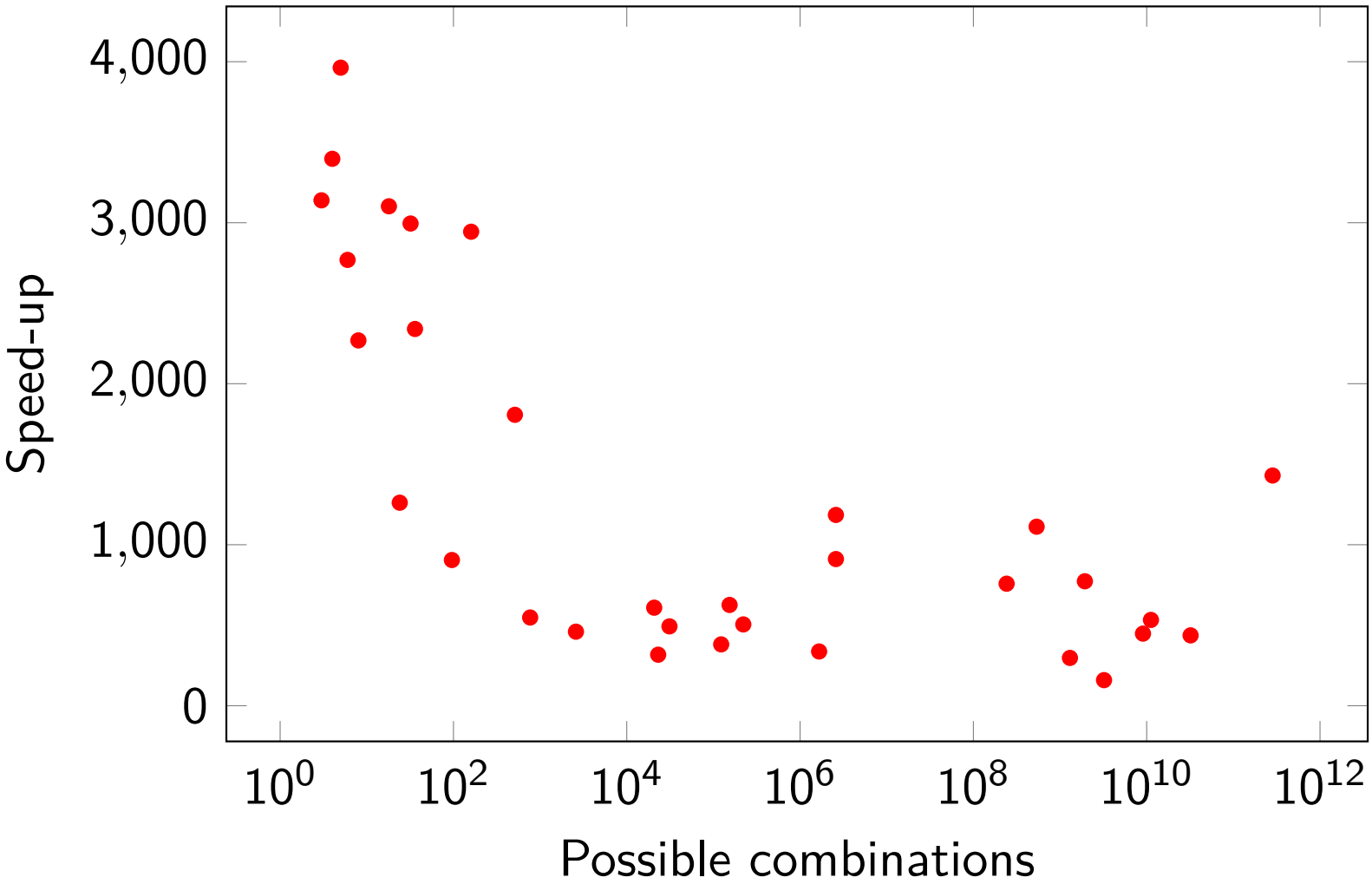
Calculations with $\varepsilon = \frac{1}{5}, \frac{1}{11}, \frac{1}{23}$, no scaling, real & complex.

Time vs. Possible Combinations



Calculations with $\varepsilon = \frac{1}{5}, \frac{1}{11}, \frac{1}{23}$, no scaling, real & complex.

PtCut vs. Satya's Solution



Compared to (Samal et al., 2016). Calculations with $\varepsilon = \frac{1}{5}$, no scaling, real. Geometric mean of factor is 920.

Program Options

Grid Sampling:

- PtCut can run calculations over a grid of parameters values. Only changed equations are re-calculated to save time.
- The grid can be additional or multiplicative, i.e., linear or factorial. This is set independent for each parameter.
- The complete 30×18 grid from (Bradford et al., 2017) takes 260 sec to compute, i.e., 0.4 sec per grid point.

Other:

- PtCut can calculate all solutions ("complex") or only the ones where the minimum is attained by monomials with opposite signs ("real"). The latter is much faster and yields fewer solutions.
- Specify $\varepsilon = \frac{1}{X}$ with `-e`X and scaling by 10^X with `-r`X.

Implementation

- PtCut is written in Python 2.7 for SageMath, a free computer algebra system.
- Tests were run on a virtual machine under Ubuntu 16.04 TLS, 64-bit, with 24 GiB of memory.
- The CPU was an Intel i7-3930K with 3.20 GHz. The code does not actively make use of multithreading.
- PtCut can be started via terminal or SSH and produces only text mode output and log/solution files.
- PtCut source code is about 3000 lines of code and licensed under LGPL v3.
- Python is easy to read and well supported by libraries, so PtCut is easy to modify.

Outlook

- What can we infer from the tropical prevariety?
- Make PtCut run independently from SageMath and possibly directly under Window, so it's easier to use.
- Complete an open implementation of an SBML-parser.
- Can we further improve the speed? E.g., Biomodel 19 with dimension 92 and $\approx 10^{60}$ combinations is still out of reach.
- Add support for variable substitution with the *vertex cover method*.
- Turn a parameter into a variable to find its breaking point without grid searching.

Thanks & References

PtCut web page: `http://wrogn.com/ptcut`

Thanks to my advisor Andreas Weber, Satya S. Samal on whose work this is based, Ovidiu Radulescu for the biology, Dima Grigoriev for the math and Jörg Zimmermann & Thomas Sturm for discussions.

Russell Bradford, James H. Davenport, Matthew England, Hassan Errami, Vladimir Gerdt, Dima Grigoriev, Charles Hoyt, Marek Košta, Ovidiu Radulescu, Thomas Sturm, and Andreas Weber. A case study on the parametric occurrence of multiple steady states. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 45–52, New York, NY, USA, 2017. ACM.

Nick I. Markevich, Jan B. Hoek, and Boris N. Kholodenko. Signaling switches and bistability arising from multisite phosphorylation in protein kinase cascades. *The Journal of Cell Biology*, 164(3):353–359, 1 2004.

Satya Swarup Samal, Aurélien Naldi, Dima Grigoriev, Andreas Weber, Nathalie Théret, and Ovidiu Radulescu. Geometric analysis of pathways dynamics: Application to versatility of TGF-β receptors. *Biosystems*, 149:3–14, 2016.