Using PtCut to Compute Tropical Equilibrations

Christoph Lüders

Universität Bonn

2 July 2018, JRC-COMBINE, RWTH Aachen

Overview

Mission Statement

- Input: a system of polynomial equations.
- Output: the tropical equilibration (or prevariety, respectively) of the system.
- The system consists of the ODEs from a (bio)chemical reaction network, with $\mathrm{d}x_i/\mathrm{d}t = 0$.
- Symbolic rate constants are OK, but for computation values have to be specified.
- We want to deduce approximate knowledge about the system from the study of its tropical solution.

Tropicalization (1)

Let f be a multivariate polynomial in $x_i \in \mathbb{R}_+$ with coefficients $k_i \in \mathbb{R}_+$. Call the monomials with positive and negative sign p_i and n_j, respectively. Furthermore, let $\varepsilon = 1/t$ with $t > 1$.

$$\sum p_i = \sum n_j$$

$$\max\{p_i\} \leq \sum p_i = \sum n_j \leq |n| \cdot \max\{n_j\}$$

$$\max\{\log_t p_i\} \leq \max\{\log_t |n| + \log_t n_j\}$$

$$-\max\{\log_t p_i\} \geq -\max\{\log_t |n| + \log_t n_j\}$$

$$\min\{-\log_t p_i\} \geq \min\{-\log_t |n| - \log_t n_j\}$$

Since $\log_t x = \frac{\ln x}{\ln(1/\varepsilon)} = \frac{\ln x}{\ln 1 - \ln \varepsilon} = -\frac{\ln x}{\ln \varepsilon} = -\log_\varepsilon x$:

$$\min\{\log_\varepsilon p_i\} \geq \min\{\log_\varepsilon |n| + \log_\varepsilon n_j\}$$

With $\lim_{\varepsilon \to 0} \log_\varepsilon |n| = 0$:

$$\min\{\log_\varepsilon p_i\} = \min\{\log_\varepsilon n_j\}$$

Tropicalization (2)

- Introduce new variable names: write $\hat{x}_i$ instead of $\log_\varepsilon x_i$.
- $\log_\varepsilon(\prod x_i^{e_i})$ can be written as $\sum \hat{x}_i e_i$.
- All Operations are lowered by one in the sequence of hyperoperations:
  - Plus and minus become minimum.
  - Because of the $\log_\varepsilon$, multiplication and division become addition and subtraction.
  - Exponentiation becomes multiplication.
- Example: the classical equation $x_1 + 3x_1x_2 + x_3^5$ becomes $\min(\hat{x}_1, \log_\varepsilon 3 + \hat{x}_1 + \hat{x}_2, 5\hat{x}_3)$ in the tropical world.
- By abuse of language, I often keep the classical variable names: $\mathrm{Trop}(x_4 + x_5^4) = \min(x_4, 4x_5)$.
- For numerical stability, the result of $\log_\varepsilon x$ is rounded to the nearest integer and possibly scaled by 10^r: $x_0 = \mathrm{round}(10^r \cdot \log_\varepsilon k)/10^r$.

Geometrical Interpretation of Tropicalized System

- Recall that $\min\{\hat{p}_i\} = \min\{\hat{n}_j\}$.
- The equation is satisfied if the minimum is attained at least twice:

$$\hat{p}_k \geq \hat{p}_i = \hat{n}_j \leq \hat{n}_\ell. \tag{1}$$

  The solution is called a *tropical root*. The equation induces a hyperplane and the inequalities induce half-spaces.
- Cycling over all pairs of monomials and applying (1) defines a *tropical hypersurface*, inducing a set of polyhedra.
- Given a set f_s of equations, the intersection of the tropical hypersurfaces of multiple equations is called a *tropical prevariety*.
- For "complex" solutions, the sign of monomials is ignored:

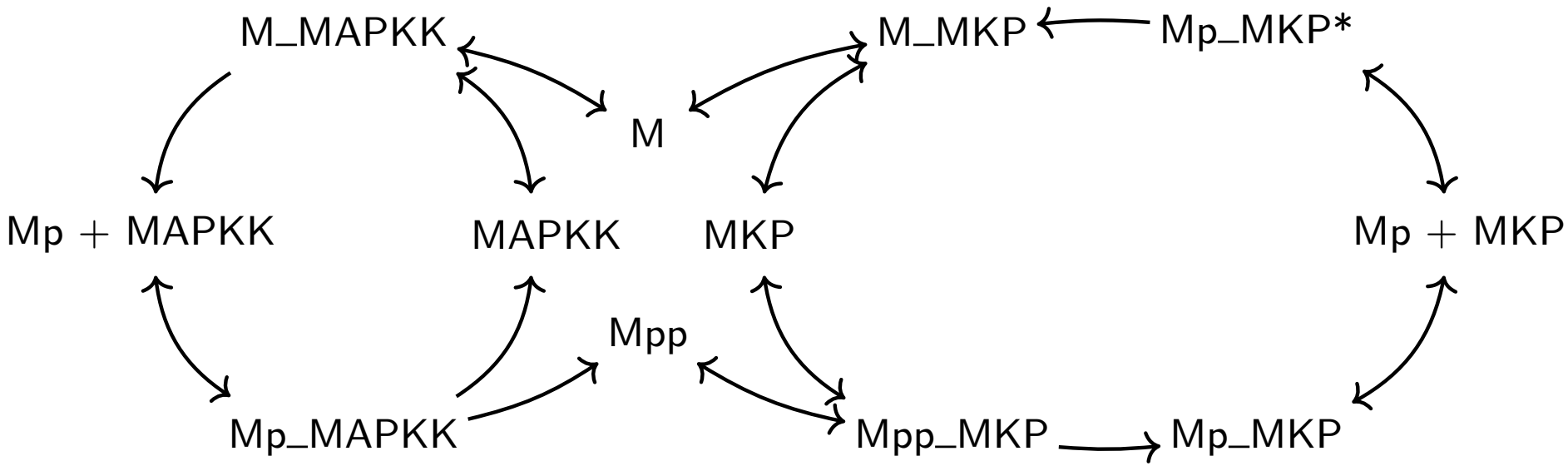
$$\hat{p}_i = \hat{p}_j \leq \hat{p}_k, \quad i \neq j.$$

PtCut Operational Overview

- Polynomial systems and rate constants are often derived from the BioModels database. An SBML-parser is needed to extract the ODEs from SBML.
- PtCut reads the polynomial system as a list of equations f_s in x_i with parameters k_j.
- Parameters are replaced, then polynomials are tropicalized (by use of a specific value for ε) and from there sets of polyhedra are created.
- These sets of polyhedra are intersected one by one to get the tropical prevariety.
- Some tricks help to lower the run-time (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 the 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 H_i that describes a set (or disjunction) of polyhedra P_{ij}:

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

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

$$U = \bigcap_i^m H_i = \bigcap_i^m \bigcup_j P_{ij}.$$

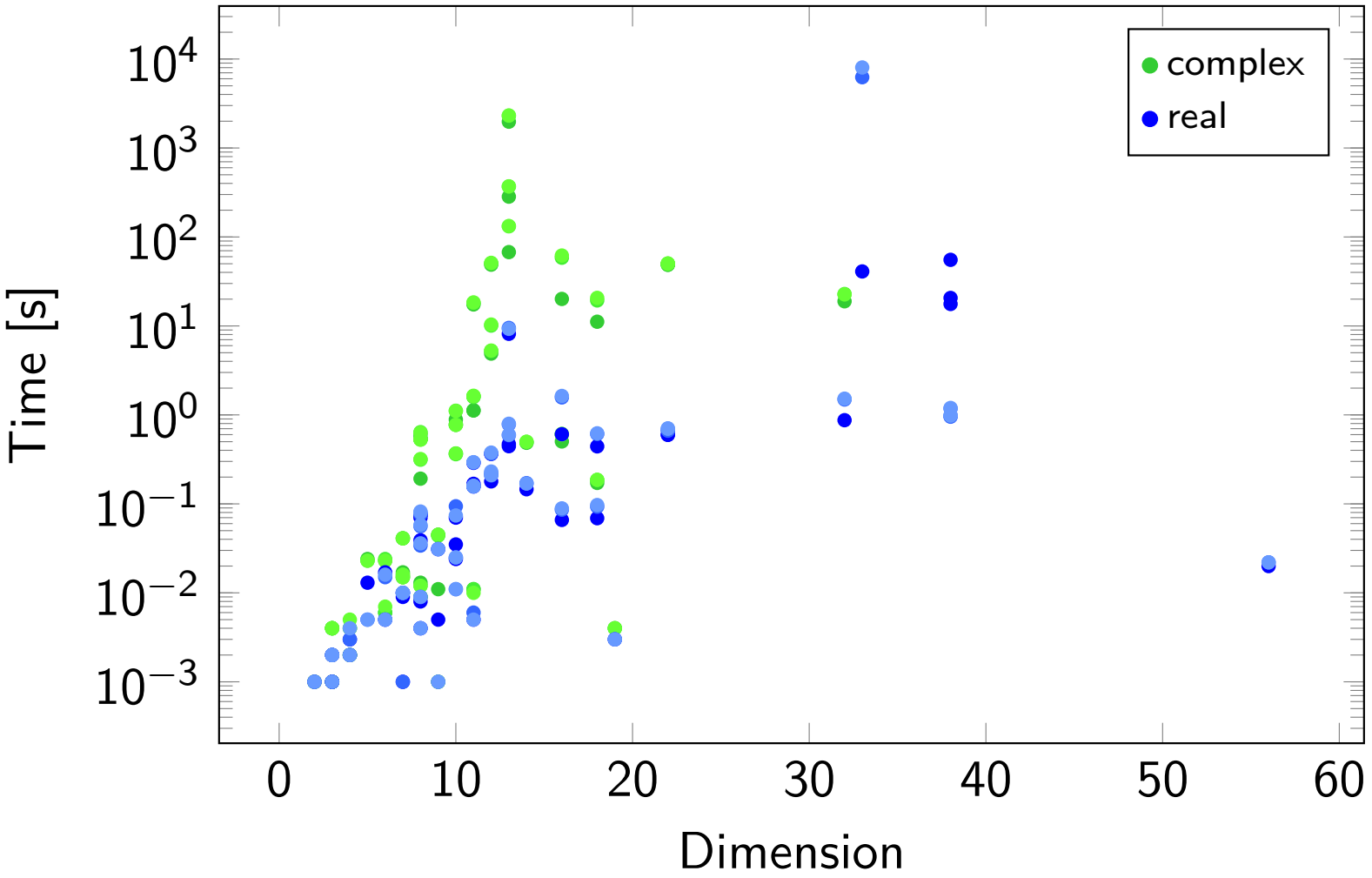
- Because of distributivity, we can rewrite that as:

$$U = \bigcup_{\alpha \in \mathcal{I}} \bigcap_i^m P_{i\alpha_i}, \quad \text{with } \mathcal{I} = \bigtimes_i^m |H_i|.$$

Combining All Polytopes (2)

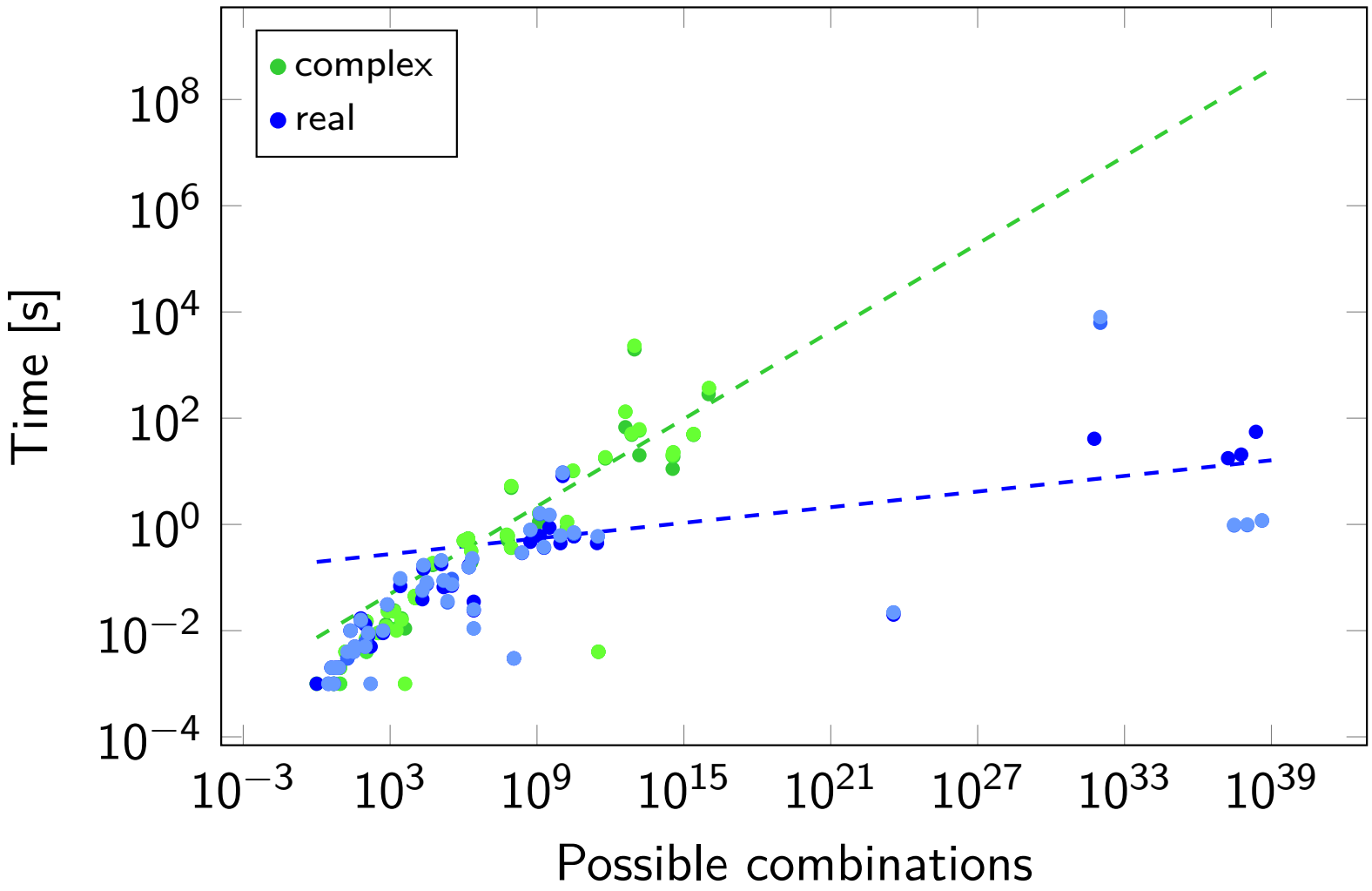
- The number of *possible* combinations is exponential in m.
- However, the biological models we examined so far have usually a very small (< 10) number of polyhedra in the prevariety, with only one exception (BioModel 102), where it is ≈ 400.
- Still, the number of *intermediate* polyhedra can be much 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.

Dimension vs. Time



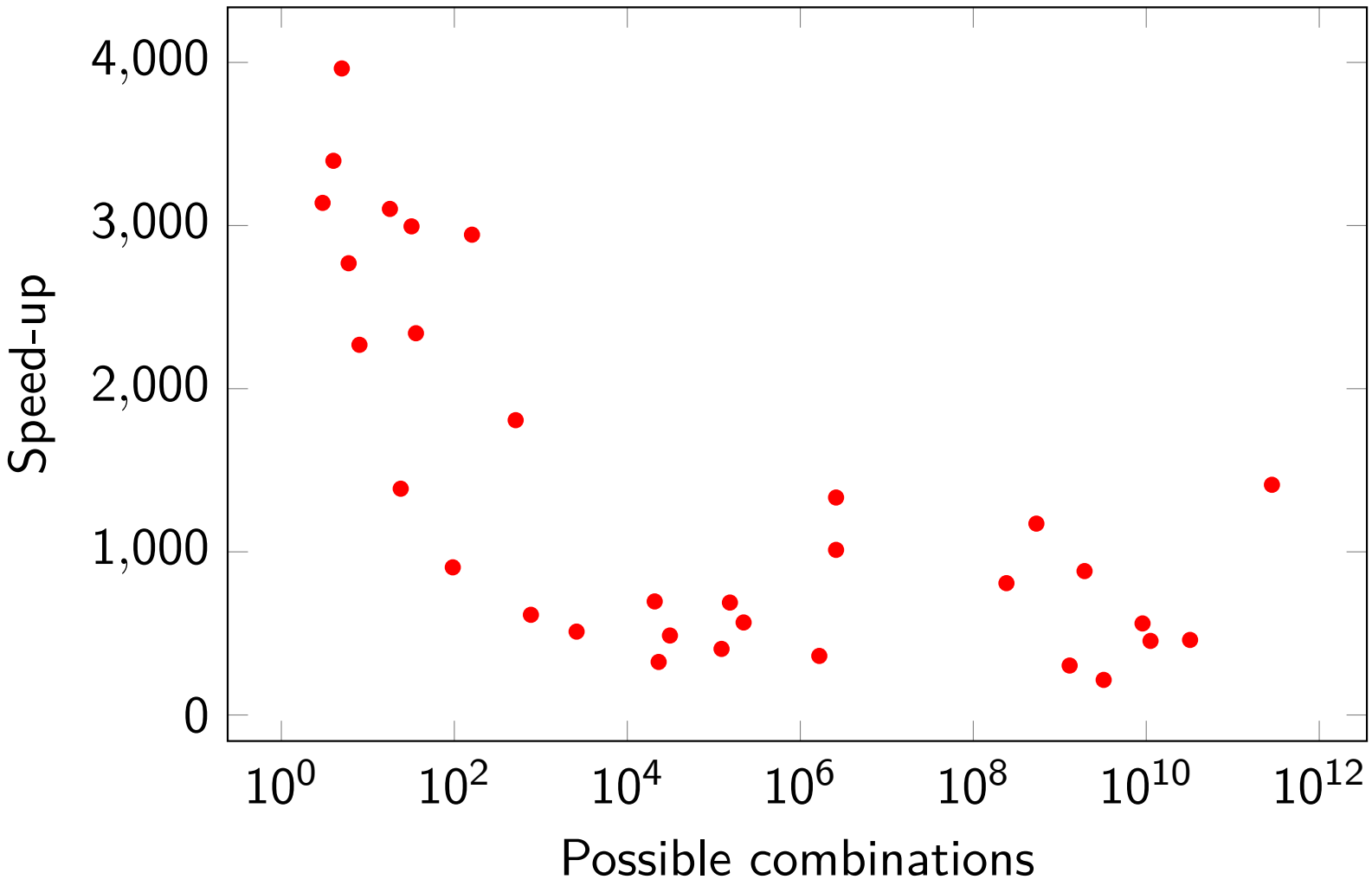
Calculations with $\varepsilon = \frac{1}{5}, \frac{1}{1.1}, \frac{1}{1.01}$, no scaling, real & complex.

Possible Combinations vs. Time



Calculations with $\varepsilon = \frac{1}{5}, \frac{1}{1.1}, \frac{1}{1.01}$, no scaling, real & complex.

PtCut vs. Samal et al. (2016)



Calculations with $\varepsilon = \frac{1}{5}$, no scaling, real. Geometric mean of factor is 968.

The Choice of ε

- Recall that $x_0 = \text{round}(10^r \cdot \log_\varepsilon k)/10^r$ with the accuracy r.
- For $\varepsilon = 1/t$, $t > 1$ it holds that

$$\log_\varepsilon k = \frac{\ln k}{\ln(1/t)} = \frac{\ln k}{\ln 1 - \ln t} = -\frac{\ln k}{\ln t},$$

  and hence

$$\lim_{\varepsilon \to 0} \log_\varepsilon k = \lim_{t \to \infty} \log_{1/t} k = 0.$$

- In order not to loose information when $\varepsilon \to 0$, we must increase the accuracy r.

The Choice of ε and Accuracy

- Assume we have ε and scaling $d = 10^r$. And assume we have another ε' with no scaling:

$$x_0 = \frac{1}{d}\,\mathrm{round}(d \cdot \log_\varepsilon k), \qquad x_0' = \mathrm{round}(\log_{\varepsilon'} k).$$

- Since it is only scaling, ignore the $\frac{1}{d}$ and get:

$$\begin{aligned} d \cdot \log_\varepsilon k &= \log_{\varepsilon'} k \\ d\frac{1}{\ln \varepsilon} &= \frac{1}{\ln \varepsilon'} \\ \ln \varepsilon' &= \frac{\ln \varepsilon}{d} \\ \varepsilon' &= e^{\ln \varepsilon \cdot \frac{1}{d}} = \varepsilon^{\frac{1}{d}} = \sqrt[d]{\varepsilon} \end{aligned}$$

- Thus, we can convert ε and $d = 10^r$ into a single "effective" ε'.

The "Effective" ε

- Some numeric examples of ε and d:

ε	d	"Effective" ε'
$1/5$	10^1	$1/1.17 = 1/(1 + 1.7 \cdot 10^{-1})$
$1/5$	10^6	$1/(1 + 1.6 \cdot 10^{-6})$
$1/29$	10^6	$1/(1 + 3.4 \cdot 10^{-6})$
$1/5$	10^{10}	$1/(1 + 1.6 \cdot 10^{-10})$
$1/10^3$	10^2	$1/1.072 = 1/(1 + 7.2 \cdot 10^{-2})$
$1/10^6$	10^4	$1/1.0014 = 1/(1 + 1.4 \cdot 10^{-3})$
$1/10^9$	10^4	$1/1.0021 = 1/(1 + 2.1 \cdot 10^{-3})$

- Using $\varepsilon = \frac{1}{t}$ is like drawing everything in a logarithmic plot with discrete "slots", each a factor of t wide.
- In practice, we should maybe aim for slots that are 10% or 1% wide, leading to values for t of 1.1 or 1.01, respectively.

Program Options

Grid Sampling:

- PtCut can run calculations over a grid of parameter values. Only changed equations are re-calculated to save time.
- The grid can be additional or multiplicative, i.e., linear or exponential. This is set independently for each parameter.
- The complete 30×18 grid from (Bradford et al., 2017) takes 162 sec to compute, i.e., 0.276 sec per grid point.

Other:

- PtCut can calculate the prevariety ("complex solutions") or only the equilibrations, where the minimum is attained by monomials with opposite signs ("real solutions"). The latter is much faster and yields fewer solutions.

Implementation

- PtCut is written for Python 2.7 & 3.x and uses PPLpy, a polyhedral library, or 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 freely available, 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 solution?
- Make PtCut run directly under Window, so it's easier to use.
- Complete implementation of a free SBML-parser.
- Reorder the intersections to lower the number of intermediate polyhedra. This has the potential to drastically reduce run-time.
- 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 (whose work I continue), Ovidiu Radulescu for biological and Dima Grigoriev for mathematical insights, Thomas Sturm for the mathematical motivation, Jörg Zimmermann for discussions and also Jonas Weinz for PPLpy wheels.

R. Bradford, J. H. Davenport, M. England, H. Errami, V. Gerdt, D. Grigoriev, C. Hoyt, M. Košta, O. Radulescu, T. Sturm, A. 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*, pages 45–52, 2017.

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

S. S. Samal, A. Naldi, D. Grigoriev, A. Weber, N. Théret, O. Radulescu. Geometric analysis of pathways dynamics: Application to versatility of TGF-β receptors. *Biosystems*, 149:3–14, 2016.