Parameter Space Decomposition of Regulatory Networks with Multiple Thresholds

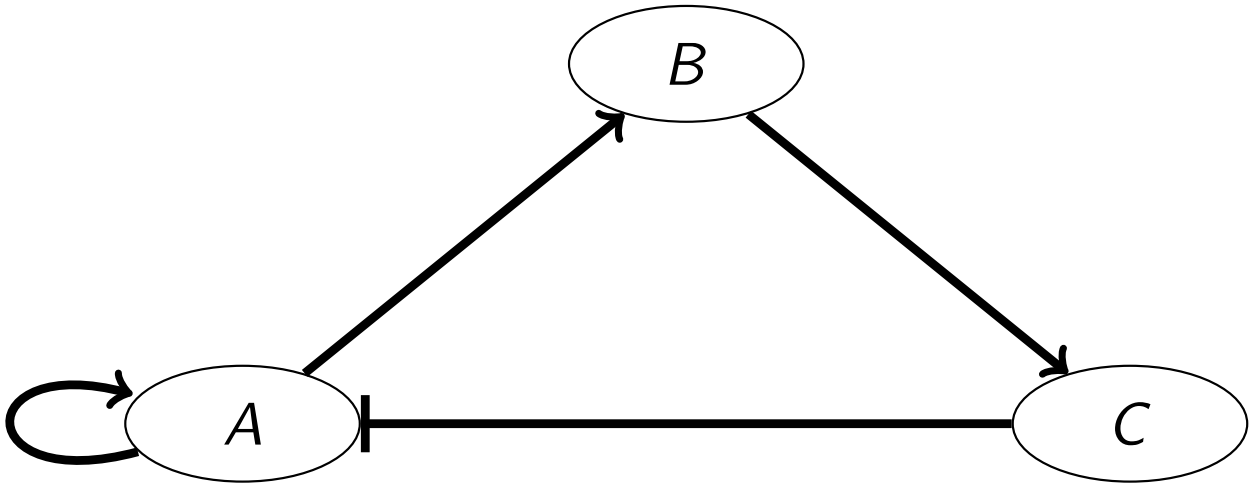
Adam Zheleznyak
Mentors: Marcio Gameiro and Konstantin Mischaikow

DIMACS REU

July 23, 2020

Regulatory Networks

Collection of species (DNA, RNA, proteins) interacting:



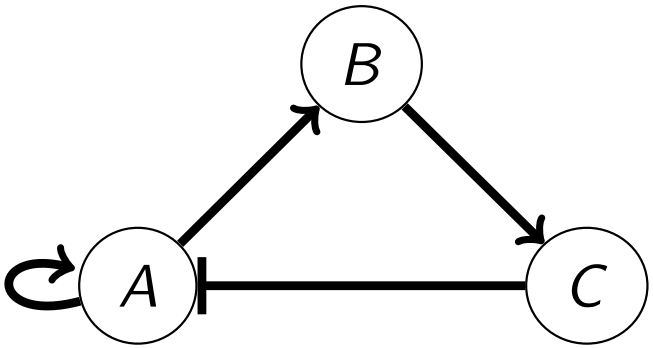
We want to understand the kinds of dynamics we can get, but this is difficult as there are many different parameters that can vary:

- How much one species affects another
- When each species starts to affect another
- The decay rate of each species

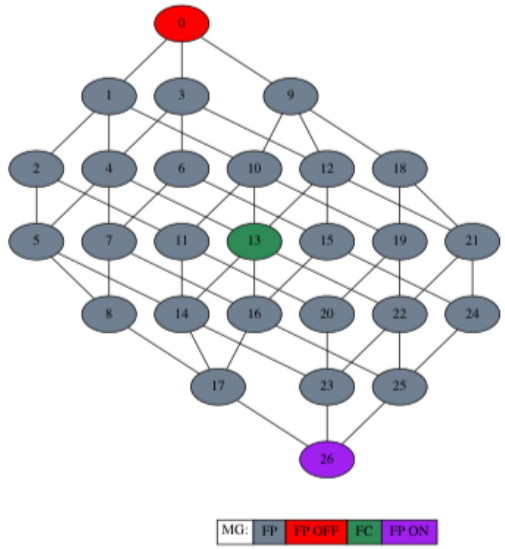
DSGRN[1]

My mentors have developed a computational technique that makes analyzing the dynamics of regulatory networks feasible and have created the software *Dynamic Signatures Generated by Regulatory Networks* (DSGRN).

In: Regulatory Network



Out: Parameter Graph



[1] Bree Cummins, Tomas Gedeon, Shaun Harker, Konstantin Mischaikow, and Kafung Mok. *Combinatorial Representation of Parameter Space for Switching Systems*. SIAM Journal on Applied Dynamical Systems, 15 (2016).

Mathematical Definition of a Regulatory Network

In DSGRN, this is how a regulatory network is modeled:

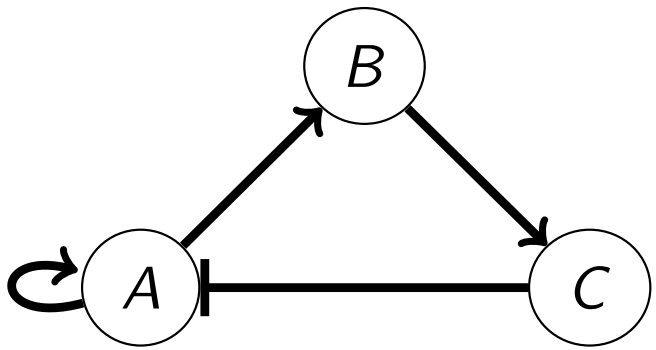
Variables: $x_1, \ldots, x_N \in \mathbb{R}_{>0}$

Parameters:

- Each species has a decay rate γ_j
- Each edge has a low value $\ell_{j,i}$, high value $\ell_{j,i} + \delta_{j,i}$, and threshold $\theta_{j,i}$
- Each $\gamma_j, \ell_{j,i}, \delta_{j,i}, \theta_{j,i} \in \mathbb{R}_{>0}$

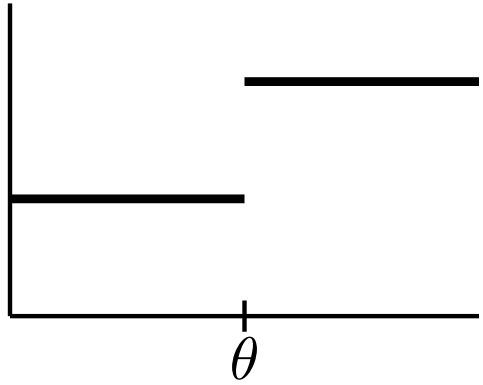
Differential equations: $\dot{x}_j = -\gamma_j + \Lambda_j(x)$, where

$$\Lambda_j(x) = \sum_{i \to j} \begin{cases} \ell_{j,i} & x_i < \theta_{j,i} \\ \ell_{j,i} + \delta_{j,i} & x_i > \theta_{j,i} \end{cases} + \sum_{i \dashv j} \begin{cases} \ell_{j,i} + \delta_{j,i} & x_i < \theta_{j,i} \\ \ell_{j,i} & x_i > \theta_{j,i} \end{cases}$$

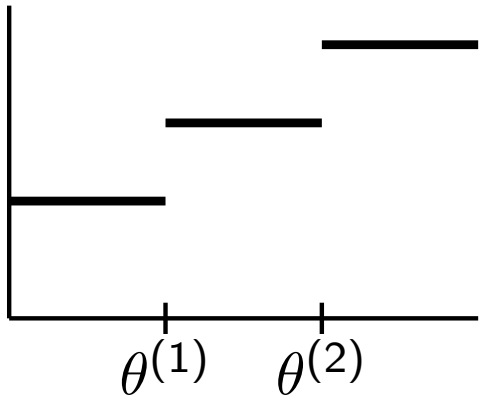


My Work

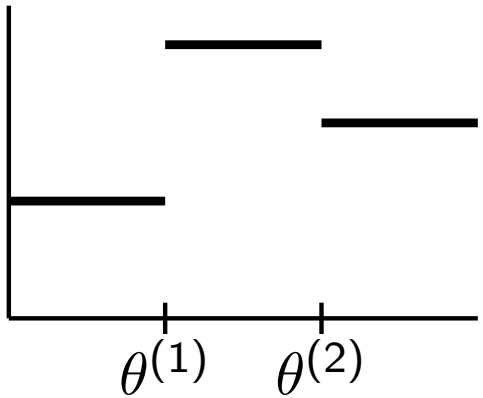
For my project, I worked on generalizing the interactions between species, allowing there to be multiple thresholds:
For some m, each edge has $\theta^{(1)}, \ldots, \theta^{(m-1)}$ as thresholds and $\ell, \delta^{(1)}, \ldots, \delta^{(m-1)}$ as the expression levels. So instead of:



we might have:



or



Parameter Space Decomposition

It turns out that the most computationally difficult part of DSGRN is figuring out what parameters are even possible (finding the dynamical signatures is actually very fast).

For example, given ℓ, δ, θ we could have:

$$\ell < \theta \text{ and } \theta < \ell + \delta$$

But we can't have:

$$\theta < \ell \text{ and } \ell + \delta < \theta$$

since this means $\ell + \delta < \ell$ which contradicts the fact that $\delta > 0$.

This is a simple example, but with more variables this quickly becomes difficult. What DSGRN does is it uses a pre-calculated database that stores this kind of information.

Finding Linear Extensions

Recall $\Lambda_j(x)$ which is defined as some sum of $\ell_{j,i}, \delta_{j,i}^{(1)}, \ldots, \delta_{j,i}^{(m-1)}$ depending on which thresholds $\theta_{j,i}^{(k)}$ are activated given each edge $i \to j$.

With n edges going into species j, there are m^n possible values for $\Lambda_j(x)$.

What we care about is the possible orderings of these values of $\Lambda_j(x)$. Because we already have some relations (e.g $\ell < \ell + \delta$ always), this is the same as finding all total orders that linearly extend a partial order. This is called the *linearly constrained linear extension problem* (LC-LEP).

For example, if $n = 2$ and $m = 2$, there are four possible inputs x_1, x_2, x_3, x_4 giving four possible values of $\Lambda_j(x_i)$. One ordering is:

$$\Lambda_j(x_1) < \Lambda_j(x_2) < \Lambda_j(x_3) < \Lambda_j(x_4)$$

But another ordering could be:

$$\Lambda_j(x_1) < \Lambda_j(x_3) < \Lambda_j(x_2) < \Lambda_j(x_4)$$

Solving LC-LEP Efficiently

As a part of my project, I implemented a recently developed algorithm for LC-LEP.[2]

Because in the cases I am looking at, $\Lambda_j(x_i)$ depends linearly on the parameters of the regulatory network, I can implement this algorithm.

Hiding most of the detail, the algorithm works by using these facts:

1. Using the linearity of $\Lambda_j(x_i)$, there is a vector that "represents" $\Lambda_j(x_i)$ for each i (by dual vector spaces).
2. Similarly, there is a representation vector for each relation in the partial ordering being extended.
3. With the representation vectors, the problem is translated into a series of linear feasibility problems, for which fast algorithms exist. Pick one.

[2] Shane Kepley, Konstantin Mischaikow, Lun Zhang. *Computing linear extensions for Boolean lattices with algebraic constraints*.

Results So Far

With the algorithm, I calculated a database of total orders.

Here, n is the number of inputs and $m-1$ is the number of thresholds.

$(n=2,\ m=2)$: 2 total orders (instant)
$(n=3,\ m=2)$: 12 total orders (instant)
$(n=4,\ m=2)$: 336 total orders ($\sim$ 90 seconds)
$(n=2,\ m=3)$: 36 total orders (instant)
$(n=2,\ m=4)$: 6660 total orders ($\sim$ 5 minutes)

Each of $(n=3,\ m=3)$ and $(n=2,\ m=5)$ cases didn't finish after 8 hours on my laptop.

Next Steps

There is still more work to be done:

- Calculate the total orders for larger n and m on a server using distributed CPUs.
- Modify the code of DSGRN so that it can handle multiple thresholds in order to use the database I've created.

Acknowledgements

This work was carried out while I was a participant in the 2020 DIMACS REU program at Rutgers University, supported by NSF HDR TRIPODS award CCF-1934924.

Thank you to Lazaros Gallos and Parker Hund for organizing the REU.

I'd also like to acknowledge the support and guidance of Marcio Gameiro, Shane Kepley, and Konstantin Mischaikow. Thank you for making the summer great!