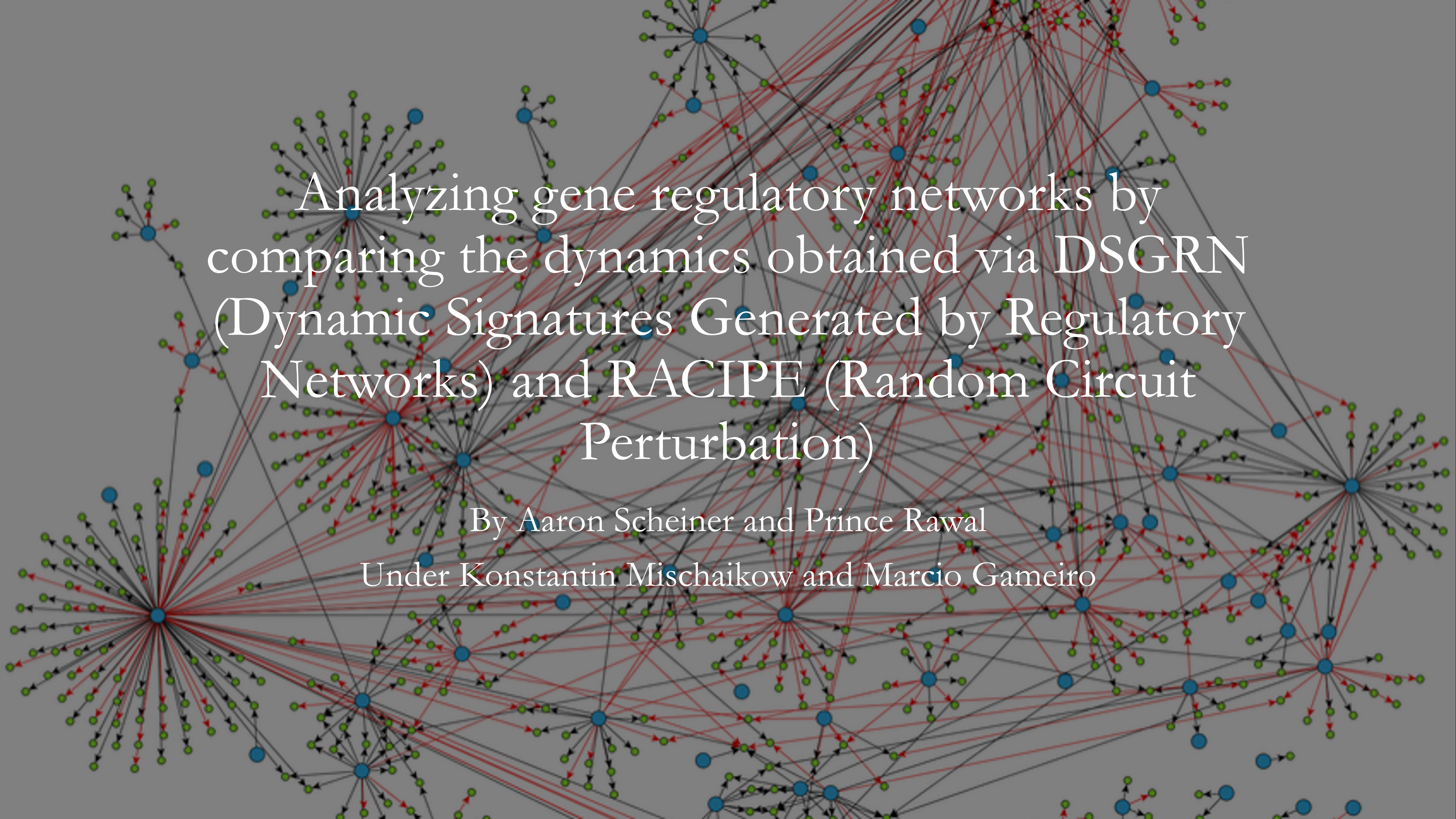
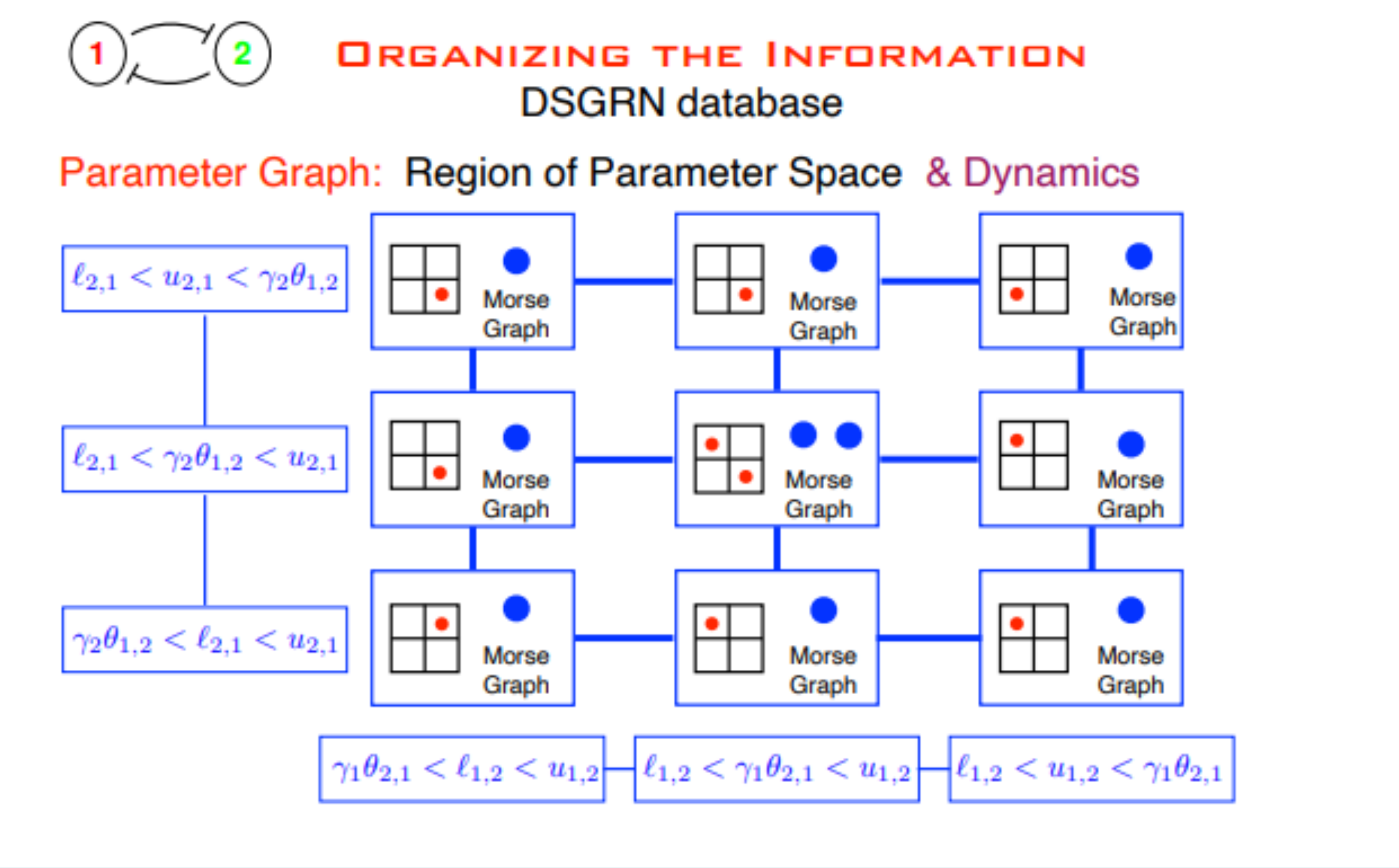


Analyzing gene regulatory networks by comparing the dynamics obtained via DSGRN (Dynamic Signatures Generated by Regulatory Networks) and RACIPE (Random Circuit Perturbation)

By Aaron Scheiner and Prince Rawal

Under Konstantin Mischaikow and Marcio Gameiro

Essential Nodes and Their Neighbors



ORGANIZING THE INFORMATION

DSGRN database

Parameter Graph: Region of Parameter Space & Dynamics

$\ell_{2,1} < u_{2,1} < \gamma_2\theta_{1,2}$

$\ell_{2,1} < \gamma_2\theta_{1,2} < u_{2,1}$

$\gamma_2\theta_{1,2} < \ell_{2,1} < u_{2,1}$

$\gamma_1\theta_{2,1} < \ell_{1,2} < u_{1,2}$

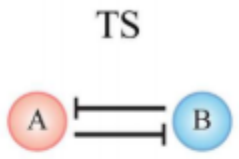
$\ell_{1,2} < \gamma_1\theta_{2,1} < u_{1,2}$

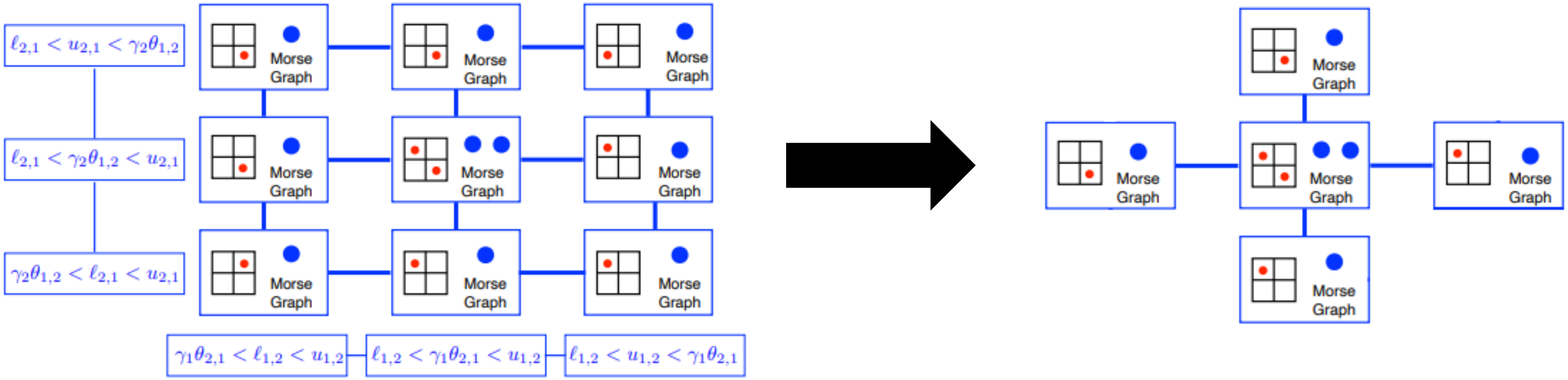
$\ell_{1,2} < u_{1,2} < \gamma_1\theta_{2,1}$

Project Progress:

This week we became more familiar with the coding side of DSGRN. I myself had to get a cluster from OARC to use DSGRN on that cluster. Later, we worked on the code provided by Marcio to run DSGRN while considering essential parameters and their neighbors. We played with DSGRN to get a better understanding of the application. Finally, we ran tests to get data from DSGRN in order to compare RACIPE and DSGRN.

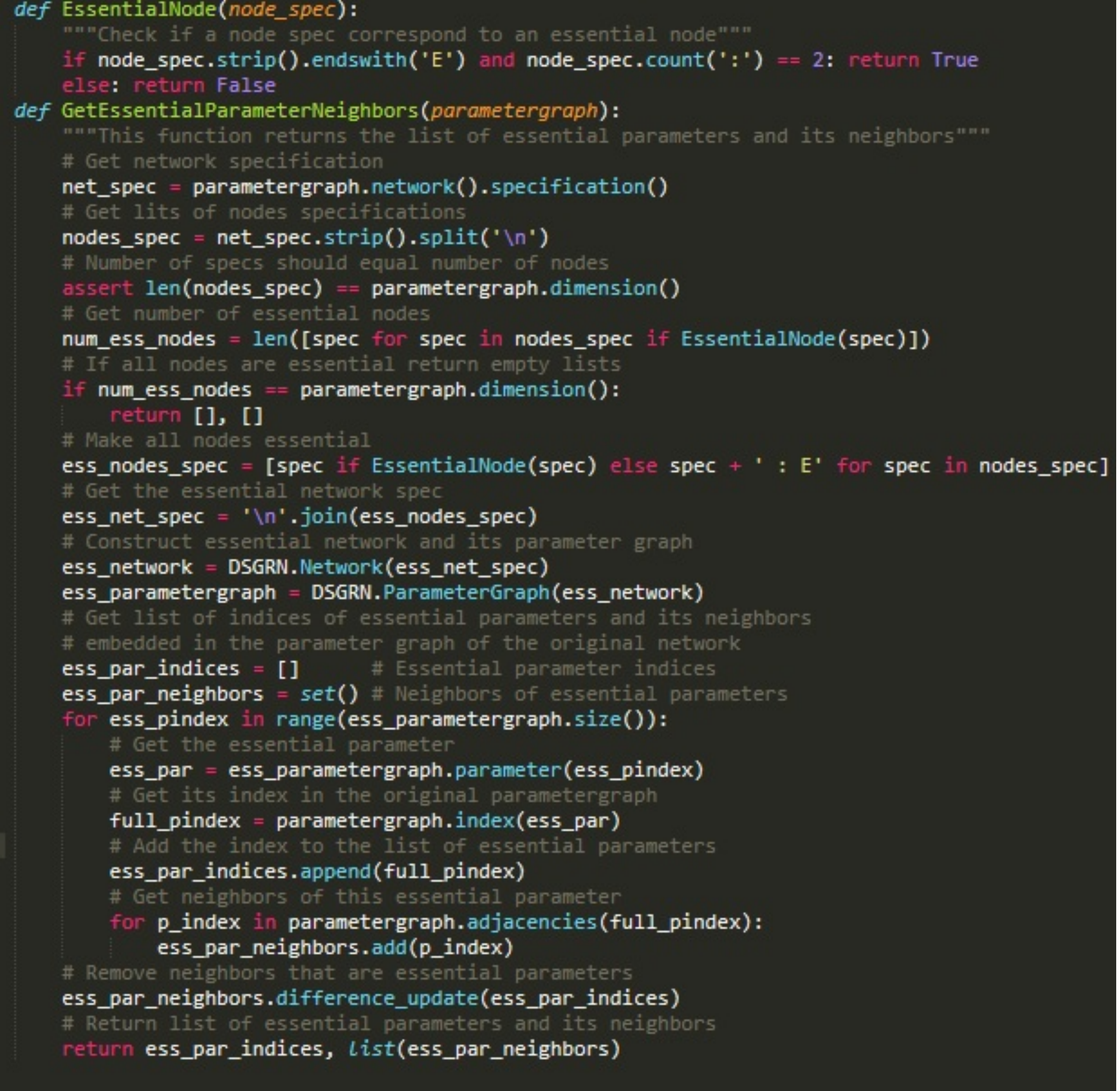
Essential Nodes and Their Neighbors:




$\ell_{2,1} < u_{2,1} < \gamma_2\theta_{1,2}$
$\ell_{2,1} < \gamma_2\theta_{1,2} < u_{2,1}$
$\gamma_2\theta_{1,2} < \ell_{2,1} < u_{2,1}$
Morse Graph
Morse Graph
Morse Graph
Morse Graph
Morse Graph
Morse Graph
Morse Graph
Morse Graph
Morse Graph
$\gamma_1\theta_{2,1} < \ell_{1,2} < u_{1,2}$
$\ell_{1,2} < \gamma_1\theta_{2,1} < u_{1,2}$
$\ell_{1,2} < u_{1,2} < \gamma_1\theta_{2,1}$
Morse Graph
Morse Graph
Morse Graph
Morse Graph
Morse Graph

- Essential nodes are where the links have probability to be both active and inactive.

- Neighbors are the 1st adjacent nodes to the essential nodes.



```python
def EssentialNode(node_spec):
    """Check if a node spec correspond to an essential node"""
    if node_spec.strip().endswith('E') and node_spec.count(':') == 2: return True
    else: return False
def GetEssentialParameterNeighbors(parametergraph):
    """This function returns the list of essential parameters and its neighbors"""
    # Get network specification
    net_spec = parametergraph.network().specification()
    # Get lits of nodes specifications
    nodes_spec = net_spec.strip().split('\n')
    # Number of specs should equal number of nodes
    assert len(nodes_spec) == parametergraph.dimension()
    # Get number of essential nodes
    num_ess_nodes = len([spec for spec in nodes_spec if EssentialNode(spec)])
    # If all nodes are essential return empty lists
    if num_ess_nodes == parametergraph.dimension():
        return [], []
    # Make all nodes essential
    ess_nodes_spec = [spec if EssentialNode(spec) else spec + ' : E' for spec in nodes_spec]
    # Get the essential network spec
    ess_net_spec = '\n'.join(ess_nodes_spec)
    # Construct essential network and its parameter graph
    ess_network = DSGRN.Network(ess_net_spec)
    ess_parametergraph = DSGRN.ParameterGraph(ess_network)
    # Get list of indices of essential parameters and its neighbors
    # embedded in the parameter graph of the original network
    ess_par_indices = []      # Essential parameter indices
    ess_par_neighbors = set() # Neighbors of essential parameters
    for ess_pindex in range(ess_parametergraph.size()):
        # Get the essential parameter
        ess_par = ess_parametergraph.parameter(ess_pindex)
        # Get its index in the original parametergraph
        full_pindex = parametergraph.index(ess_par)
        # Add the index to the list of essential parameters
        ess_par_indices.append(full_pindex)
        # Get neighbors of this essential parameter
        for p_index in parametergraph.adjacencies(full_pindex):
            ess_par_neighbors.add(p_index)
    # Remove neighbors that are essential parameters
    ess_par_neighbors.difference_update(ess_par_indices)
    # Return list of essential parameters and its neighbors
    return ess_par_indices, list(ess_par_neighbors)
```

Getting the Essential Parameters and Their Neighbors:

- Construct the essential network and its parameter graph.

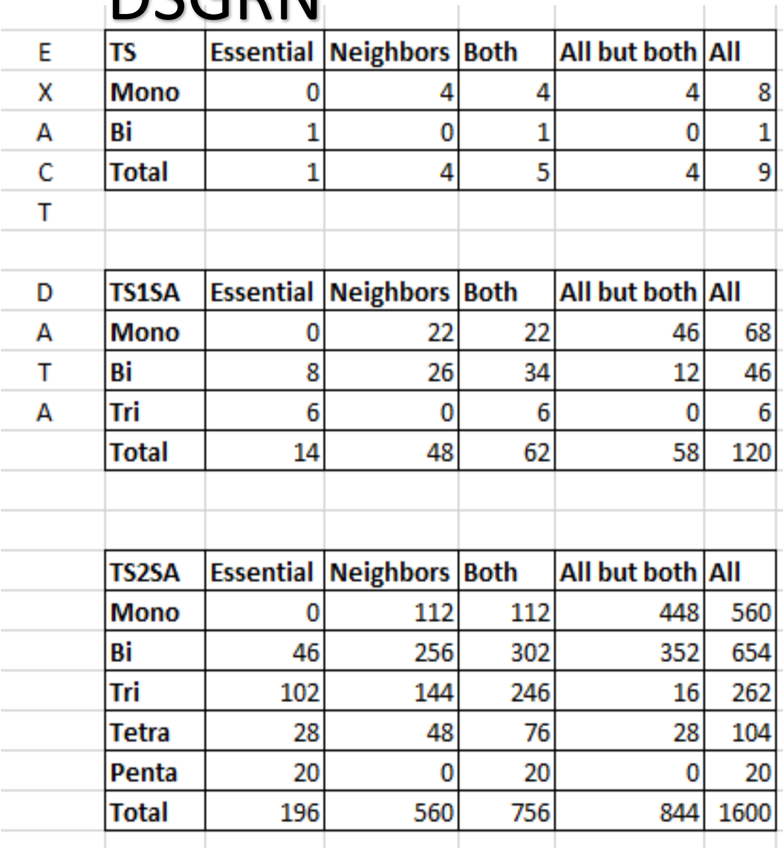
- Get the indices of essential parameters and its neighbors.

- Find the essential nodes and their indices in the original parameter graph.

- Similarly, get the neighbors and remove the neighbors that are also essential nodes.

Results for Toggles Switches:

DSGRN

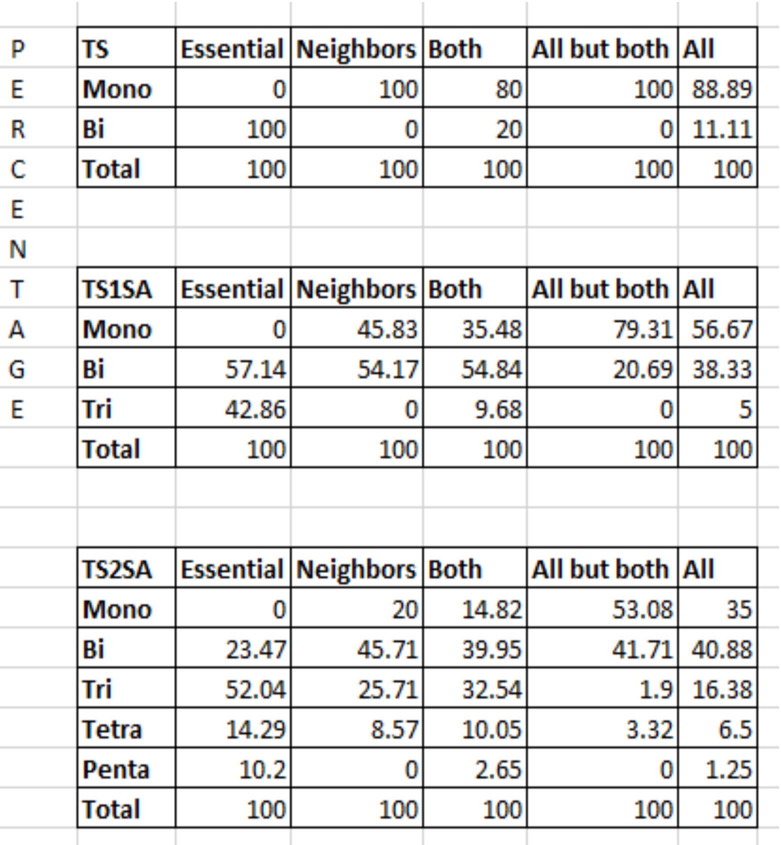


EXACT DATA

TS	Essential	Neighbors	Both	All but both	All
Mono	0	4	4	4	8
Bi	1	0	1	0	1
Total	1	4	5	4	9

TS1SA	Essential	Neighbors	Both	All but both	All
Mono	0	22	22	46	68
Bi	8	26	34	12	46
Tri	6	0	6	0	6
Total	14	48	62	58	120

TS2SA	Essential	Neighbors	Both	All but both	All
Mono	0	112	112	448	560
Bi	46	256	302	352	654
Tri	102	144	246	16	262
Tetra	28	48	76	28	104
Penta	20	0	20	0	20
Total	196	560	756	844	1600

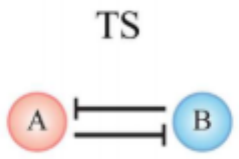


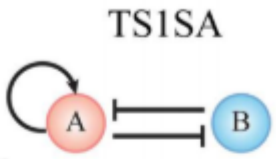
PERCENTAGE

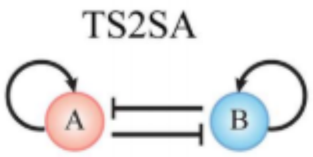
TS	Essential	Neighbors	Both	All but both	All
Mono	0	100	80	100	88.89
Bi	100	0	20	0	11.11
Total	100	100	100	100	100

TS1SA	Essential	Neighbors	Both	All but both	All
Mono	0	45.83	35.48	79.31	56.67
Bi	57.14	54.17	54.84	20.69	38.33
Tri	42.86	0	9.68	0	5
Total	100	100	100	100	100

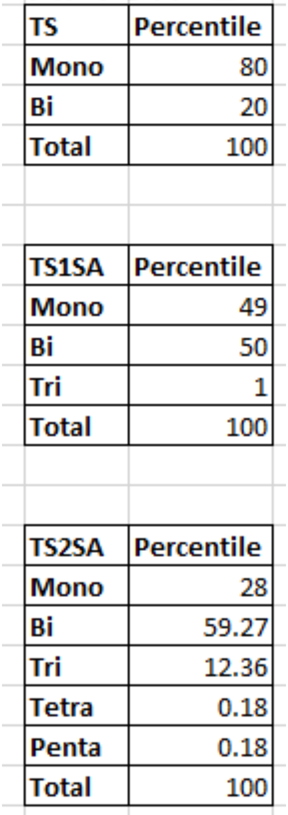
TS2SA	Essential	Neighbors	Both	All but both	All
Mono	0	20	14.82	53.08	35
Bi	23.47	45.71	39.95	41.71	40.88
Tri	52.04	25.71	32.54	1.9	16.38
Tetra	14.29	8.57	10.05	3.32	6.5
Penta	10.2	0	2.65	0	1.25
Total	100	100	100	100	100







RACIPE



TS	Percentile
Mono	80
Bi	20
Total	100

TS1SA	Percentile
Mono	49
Bi	50
Tri	1
Total	100

TS2SA	Percentile
Mono	28
Bi	59.27
Tri	12.36
Tetra	0.18
Penta	0.18
Total	100

Next Steps:

Our next step will be to see if we can figure out the possible reasons for the difference between the data generated by both applications. We will try to run RACIPE and DSGRN on some other basic models to get a better understanding of what is going on. Also, we can try to figure out if the half-functional rule is the reason for these differences and see if essential nodes and their neighbors are the right method for reproducing the half-functional rule, to replicate RACIPE's sampling methods, in DSGRN.

Thank You for Listening!

and

Thank you to the 2020 David and Dorothy Bernstein Endowed Scholarship for Summer Research and Aresty Research Center for supporting our research this summer!