

# Workshop on Genetic Regulatory Networks ISCV Valparaíso, Chile

## Numerical Study of Boolean Networks\*

G. Hernandez<sup>1</sup> and A. Moreira<sup>2</sup>

<sup>1</sup>UCHILE Center for Mathematical Modeling, Santiago, Chile

<sup>2</sup>USM Department of Informatics, Valparaiso, Chile

**\*Supported by MORPHEX Research Grant**

# Introduction: Boolean Networks Numerical Dynamics

- The objective of this talk is to present the first version of the library: Boolean Networks Numerical Dynamics, that was designed to study computationally the dynamics of boolean functions:

- Define a boolean function (threshold function):

$$f(x) = \left( f_i(x) \right)_{i=1}^n \Rightarrow f_i(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^n w_{ij} x_j - \theta_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Define different deterministic update schema:

1	1	1	1	←	Synchronous	} Block Iteration Modes
1	2	3	4	←	Asynchronous	
1	1	2	2	←	Two blocks: 1,2 ; 3,4	
1	2	2	3	←	Three blocks: 1 ; 2,3 ; 4	

# Introduction: Boolean Networks Numerical Dynamics

- Numerical Dynamics of Boolean Networks:
  - For a specific block iteration mode compute all the attractors over the hypercube
  - For all block iteration modes compute all the attractors
  - Visualizes the dynamics for an initial condition
- This library was programmed in Matlab<sup>®</sup> and utilizes some commands of the Random Boolean Network Toolbox:  
Christian Schwarzer  
Logic System Laboratory  
Swiss Federal Institute of Technology in Lausanne  
<http://www.teuscher-research.ch/rbntoolbox/UsingToolbox/gettingStarted.htm>

# Random Boolean Network Toolbox<sup>®</sup>

```
function [node,conn,rule,attractors_length,attractors_states] = using_rbnt(n,k,tsteps,mode)
%Script to use the main functions of the RBN Toolbox
%Input:
%n: number of nodes
%k: average number of inputs per node
%tsteps: number of iterations (updates)
%mode of iteration:
%CRBN (synchronous deterministic), ARBN (asynchronous non-%deterministic)
%DARBN (deterministic asynchronous), GARBN (semi-synchronous, non-deterministic),
%DGARBN (semi-synchronous, %deterministic)
%Output
%node: structure (node.state, node.nextState, node.nbUpdates, node.input, node.output,
%conn connectivity matrix defined as:
%conn(i,j)=1 if node j is output of node i
%conn(i,j)=1 if node i is input of node j
%rule: boolean rules matrix of size nx(2^k)
```

# Random Boolean Network Toolbox<sup>®</sup>

```
function [node,conn,rule,attractors_length,attractors_states] =  
using_rbnt(n,k,tsteps,mode)  
  
%Build and show the random boolean network  
[node, conn, rule] = bsn(n,k,'arrow');  
  
%Computing and visualizing the network's evolution  
[node, tsm] = displayEvolution(node,tsteps,mode);  
  
%Find the attractor in tsm  
[attrLength, attrStates] = findAttractor(tsm);  
  
%Visualizing dynamics, node statistics  
Number_changes_per_node = displayNodeStats(node,tsm)
```

# Random Boolean Network Toolbox<sup>®</sup>

```
function [node,conn,rule,attractors_length,attractors_states] = using_rbnt(n,k,tsteps,mode)
%Define new connectivity matrix new_conn
new_conn = conn;
for p=1:(round(n/2))
    j=randint(1,1,[1 n]);
    new_conn(1:n,j)=random_inputs(n,k);
end

%To change input nodes in new_node
new_node = assocNeighbours(node,new_conn);

for i=1:n
    conn_out=[];
    for j=1:n
        if (new_conn(i,j) == 1) conn_out=[conn_out j]; end
    end
    new_node(i).output=conn_out;
end

%To change boolean rules
new_rule=rule; new_rule = randint((2^k),n); new_node = assocRules(new_node,new_rule);
```

# Boolean Networks Numerical Dynamics

- `attractors_arabidopsis.m`
- `attractors_tfunction.m`
- `transition_state_matrix_arabidopsis.m`
- `transition_state_matrix_tfunction.m`
- `block_iteration_modes.m`
- `random_partition.m`
- `generating_partitions.m` (`bell_numbers.m`)
- `transition_function_arabidopsis.m`
- `transition_function_definition.m`
- `hypercube.m`

# Boolean Networks Numerical Dynamics

- `attractors_arabidopsis.m`  
%Computing the attractors of the arabidopsis over the hypercube  
%For the mode of iteration: mode ('s' asynchronous, 'p' synchronous)
- `attractors_tfunction.m`  
%Computing the attractors of a boolean threshold function  
%over the hypercube for the random mode of iteration p
- `transition_state_matrix_tfunction.m`  
%Transition state matrix for a boolean threshold function
- `block_iteration_modes.m`  
%Generates block iteration modes for partition p

# Boolean Networks Numerical Dynamics

- `block_iteration_modes.m`:

```
>> p=random_partition(8)
```

```
p =
```

```
1  2  2  3  1  2  1  4
```

```
>> bim = block_iteration_modes(8,p)
```

```
bim =
```

```
1  0  0  0  1  0  1  0
0  1  1  0  0  1  0  0
0  0  0  1  0  0  0  0
0  0  0  0  0  0  0  1
```

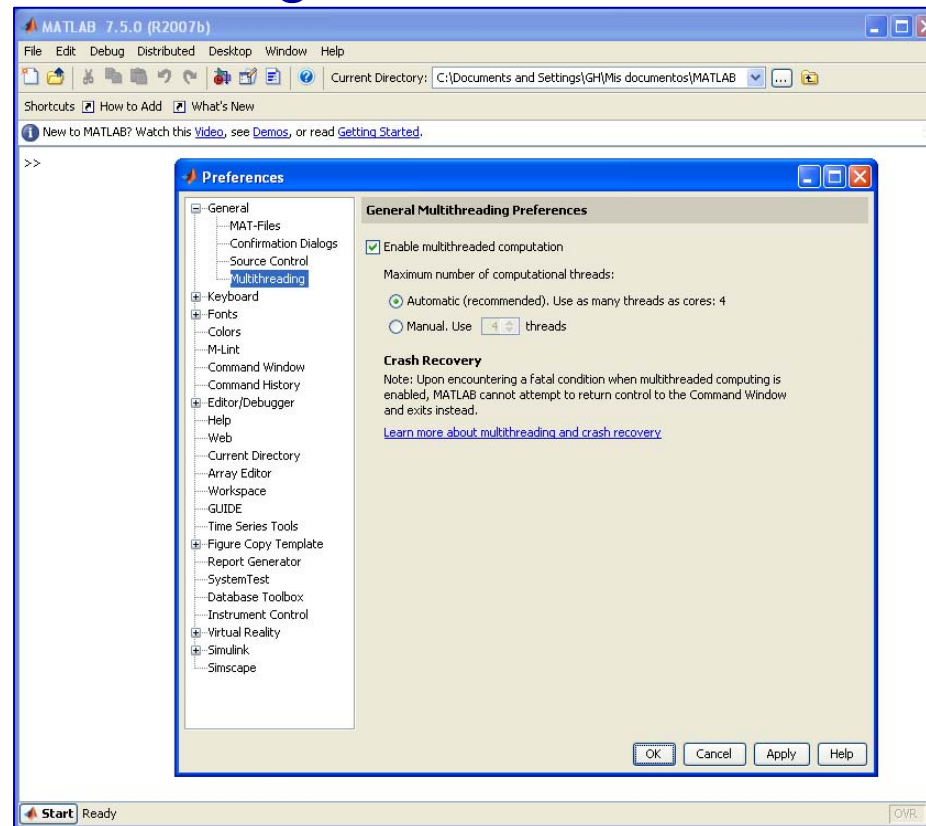
## Specific and Complete Numerical Study

- For a boolean threshold function determine all the attractors over the hypercube for a specific mode of iteration.
- For a boolean threshold function determine all the attractors over the hypercube for all the modes of iteration.
- Number of modes of iteration: Bell numbers
- Exponential growth of cputime of generating\_partitions

N	Modes of Iteration	cputime [s]
4	15	0
5	52	0
6	203	0.016
7	877	0.047
8	4140	0.23
9	21147	1.55
10	115975	14.09
11	678570	145.17
12	4213597	Out of memory

# How to improve performance ?

## ■ Matlab multithreading



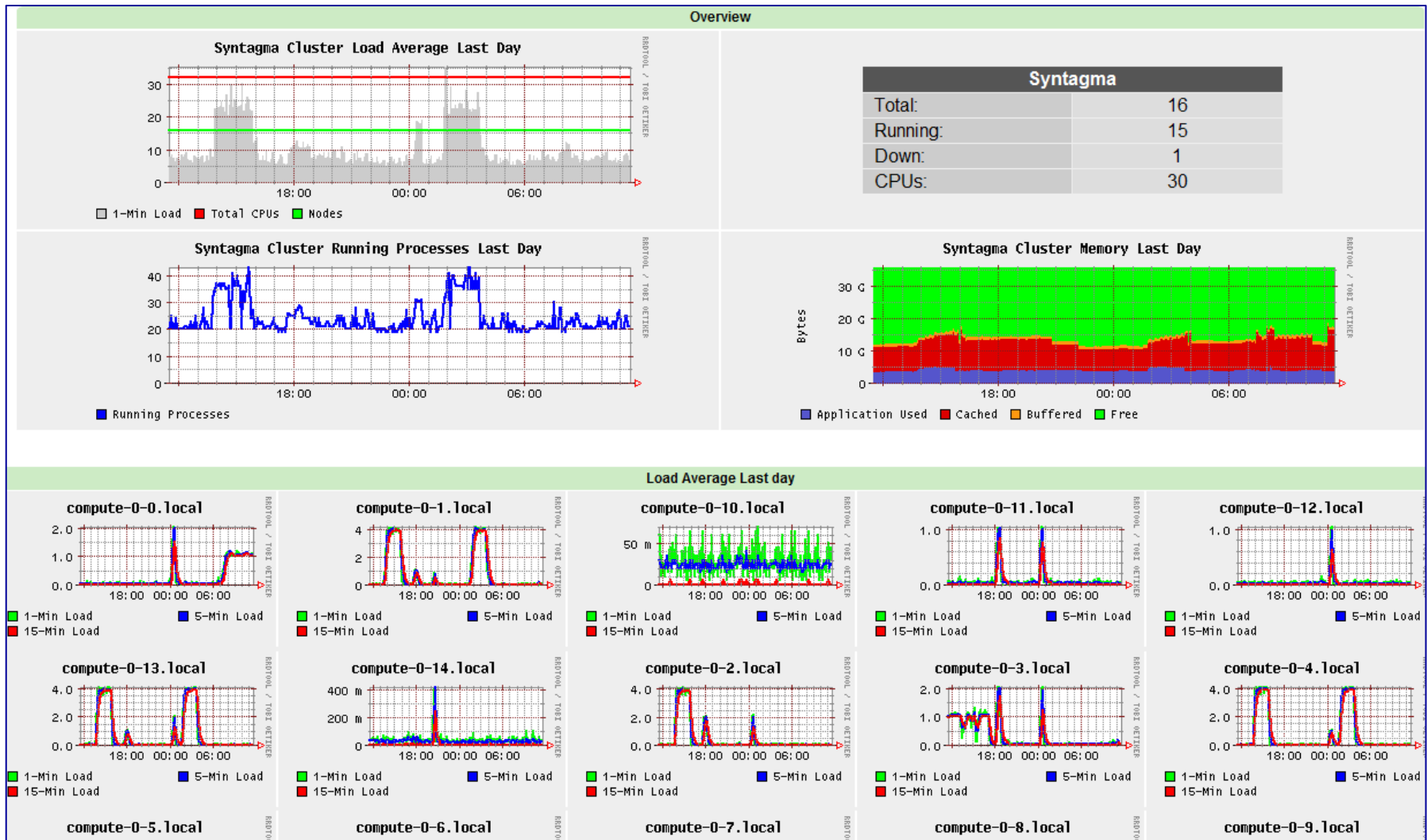
## ■ StarP: [www.interactivesupercomputing.com/products/](http://www.interactivesupercomputing.com/products/)

# How to improve performance ?

- Rewrite program in C/C++
  - Cluster computing: OpenMP and MPI Libraries
- Syntaxma (Center for Mathematical Modeling)



# Syntaxma (Center for Mathematical Modeling)



## How to improve performance ?

- Submission of process: Via Sun Grid Engine (similar to OpenPBS, but with more functionality)
- For instance, to execute `attractors_tfunction.m`  
`jattractors_tfunction.sh`  
`#!/bin/bash`  
`#$ -cwd`  
`#$ -P cmm`  
`#$ -j y`  
`#$ -m abes`  
`#$ -N attractors_tfunction`  
`#$ -M gjho@vtr.net`  
`#$ -notify`  
`#$ -S /bin/bash`  
`/opt/matlab/bin/matlab -nosplash -r`  
`[attr_length,attr_states]=attractors_tfunction -logfile attractors_tfunction.log`

# Current Research

- Work in progress:
- Compute extensive numerical simulations to study the dynamics of boolean functions.
- The simulations will be developed in three stages:
  - Medium scale sequential simulations performed on pc.
  - Medium scale parallel and distributed simulations performed on pc using multicore architectures and programming.
  - Large scale parallel simulations performed on clusters the MPI and OpenMP libraries.