Modeling organization of spiking neural nets in DEVS for high performance parallel simulation

Bernard P. Zeigler[1] & Alexandre Muzy[2]

[1]RTSync, University of Arizona – USA
[2]I3S CNRS/UNS, Sophia-Antipolis – France

April 14, 2016

Introduction

- Discrete events already used for the modeling and simulation of neural nets (Brette *et al.* , 2007).
- With DEVS: Original neuron models (Zeigler, 2005) or to abstract neural nets (Zeigler, 1975).
- *But* no DEVS neural nets dealing with stochasticity such as Markov chains or random graphs.

Introduction

- Discrete events already used for the modeling and simulation of neural nets (Brette *et al.* , 2007).
- With DEVS: Original neuron models (Zeigler, 2005) or to abstract neural nets (Zeigler, 1975).
- *But* no DEVS neural nets dealing with stochasticity such as Markov chains or random graphs.

Introduction

- Discrete events already used for the modeling and simulation of neural nets (Brette *et al.* , 2007).
- With DEVS: Original neuron models (Zeigler, 2005) or to abstract neural nets (Zeigler, 1975).
- *But* no DEVS neural nets dealing with stochasticity such as Markov chains or random graphs.

Goals

- Markov model components as DEVS pseudorandom systems
- Markov matrix models to predict the speed-up of parallel simulations
- Application to symmetric multiprocessor (SMP)
- Markov matrix model of the P-DEVS protocol

Goals

- Markov model components as DEVS pseudorandom systems
- Markov matrix models to predict the speed-up of parallel simulations
- Application to symmetric multiprocessor (SMP)
- Markov matrix model of the P-DEVS protocol

Goals

- Markov model components as DEVS pseudorandom systems
- Markov matrix models to predict the speed-up of parallel simulations
- Application to symmetric multiprocessor (SMP)
- Markov matrix model of the P-DEVS protocol

Goals

- Markov model components as DEVS pseudorandom systems
- Markov matrix models to predict the speed-up of parallel simulations
- Application to symmetric multiprocessor (SMP)
- Markov matrix model of the P-DEVS protocol

Neuron dynamic model

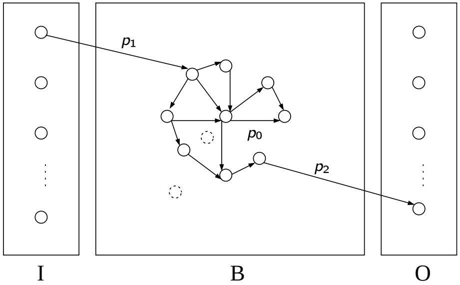
Usual *leaky integrate-and-fire model*, at time $t \in \mathbb{N}$, the *membrane potential* $s(t) \in \mathbb{R}$ of a neuron consists of:

$$s(t) = \begin{cases} rs(t-1) + \Sigma_{j=1}^{m} w_j x_j(t) & \textit{if } s(t-1) < \tau \\ 0 & \textit{otherwise} \end{cases}$$

with $r \in [0, 1]$ the *remaining coefficient*, w_j the *synaptic weight* from neuron j.
Spike emission $x(t)$ depends on *threshold* $\tau \in \mathbb{R}^+$:

$$x(t) = \begin{cases} 1 & \textit{if } s(t-1) \geq \tau \\ 0 & \textit{otherwise} \end{cases}$$

Neuron graph model



Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Simulation parameters
  - probabilities: $p_1 = p_2 = 0.9$, $p_3 = 0.5$, p_0: variations,
  - all *high activity* neurons: $a = r = 1$, each threshold $\tau_i \sim \mathcal{N}(m, \sigma^2)$, with $m = 20$ and $\sigma = 1$
  - Generator period: *1*
  - No internal couplings inside input and output layers
- Parallel implementation
- Multithreaded SMP machine (80 physical cores, 160 logical ones)
  - Each coordinator with own parallelized scheduler
  - DEVSJava extension

Parallel simulation

- Variations of:
  - Number of threads (1,5,10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
  - probability p_0 (internal coupling of middle layer)
  - Number of neurons in each layer (1, 5, 10, 100, 500)
- Model each coupled model execution time reduction

Parallel simulation

- Variations of:
  - Number of threads (1,5,10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
  - probability p_0 (internal coupling of middle layer)
  - Number of neurons in each layer (1, 5, 10, 100, 500)
- Model each coupled model execution time reduction

Parallel simulation

- Variations of:
  - Number of threads (1,5,10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
  - probability p_0 (internal coupling of middle layer)
  - Number of neurons in each layer (1, 5, 10, 100, 500)
- Model each coupled model execution time reduction

Parallel simulation

- Variations of:
  - Number of threads (1,5,10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
  - probability p_0 (internal coupling of middle layer)
  - Number of neurons in each layer (1, 5, 10, 100, 500)
- Model each coupled model execution time reduction

Parallel simulation

- Variations of:
  - Number of threads (1,5,10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160)
  - probability p_0 (internal coupling of middle layer)
  - Number of neurons in each layer (1, 5, 10, 100, 500)
- Model each coupled model execution time reduction

Preliminary results

Input layer (simple stochastic generators)

Nb of neurons in middle layer	Elapsed time one thread	Best time	Speed-up ratio
1	0,520717	0,420232	1,2391179158
5	0,495051	0,495051	1
10	0,52735	0,52735	1
100	0,661288	0,661288	1
500	5,65755	5,65755	1

Rk: Each result is replicated 20 times with good confidence interval

Preliminary results

- Middle layer (send/receive neurons with int. couplings)
- gain = proba*NbNeurons

Proba	0,1			
Nb of neurons in middle layer	Elapsed time one thread	Best time	Speed-up ratio	proba*NbNeurons
1	-	-	-	0,1
5	0,369873	0,369873	1	0,5
10	0,326683	0,326683	1	1
100	1,77223	1,352	1,3108210059	10
500	44,4226	25,8828	1,7162980821	50

Proba	0,9			
Nb of neurons in middle layer	Elapsed time one thread	Best time	Speed-up ratio	proba*NbNeurons
1	-	-	-	0,9
5	0,400321	0,400321	1	4,5
10	0,403413	0,403413	1	9
100	2,7502	1,58617	1,7338620703	90
500	75,207	35,8044	2,1004960284	450

Preliminary results

Output layer (simple receiving neurons)

Nb of neurons in middle layer	Elapsed time one thread	Best time	Speed-up ratio
1	-	-	-
5	0,089175	0,089175	1
10	0,122071	0,122071	1
100	1,25386	0,860517	1,4571007894
500	32,2965	13,4208	2,4064511803

Sum-up

All layers

P0=0,1					P0=0,9		
Neurons	Input	middle	output		input	middle	output
5	0,50	0,37	0,89		0,45	0,40	0,16
10	0,53	0,33	0,12		0,49	0,40	0,18
100	0,66	1,77	1,25		0,52	2,75	1,31
500	5,66	44,42	32,30		6,12	75,21	32,44

Conclusion

- Simulations with more overhead are running
- First model of random graph based coupled models run in parallel
- Next steps [test composition of Markov models (of each layer)]:
    - Collect new results and achieve better models
    - Account for negative weights percentage for the gain (proportional)

References

Brette, Romain, Rudolph, Michelle, Carnevale, Ted, Hines, Michael, Beeman, David, Bower, JamesM., Diesmann, Markus, Morrison, Abigail, Goodman, PhilipH., Harris, Frederick C., Jr., Zirpe, Milind, Natschlager, Thomas, Pecevski, Dejan, Ermentrout, Bard, Djurfeldt, Mikael, Lansner, Anders, Rochel, Olivier, Vieville, Thierry, Muller, Eilif, Davison, Andrew P., El Boustani, Sami, & Destexhe, Alain. 2007. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of computational neuroscience*, **23**(3), 349–398.

Zeigler, Bernard P. 1975. Statistical simplification of neural nets. *International journal of man-machine studies*, **7**(3), 371–393.

Zeigler, B.P. 2005. Discrete event abstraction: an emerging paradigm for modeling complex adaptative system.