Simulating Apoptosis Using Discrete Methods: a Membrane System and a Stochastic Approach

John Jack,[1] Francisco J. Romero-Campero,[2] Mario J. Pérez-Jiménez,[2]
Oscar H. Ibarra[3] and Andrei Păun[1,4†]

[1] Department of Computer Science/IfM, Louisiana Tech University,
P.O. Box 10348, Ruston, LA 71272, USA {johnjack, apaun}@latech.edu
[2] Department of Computer Science and Artificial Intelligence, University of Sevilla,
Avda. Reina Mercedes s/n 41012, Sevilla, Spain {fran, marper}@us.es
[3] Department of Computer Science, University of California - Santa Barbara,
Santa Barbara, CA 93106, USA ibarra@cs.ucsb.edu
[4] Universidad Politécnica de Madrid - UPM, Facultad de Informática
Campus de Montegancedo S/N, Boadilla del Monte, 28660 Madrid, Spain

Abstract. Membrane Systems provide an intriguing method for modeling biological systems at a molecular level. The hierarchical structure of Membrane Systems lends itself readily to mimic the nature and behavior of cells. We have refined a technique for modeling the type I and type II FAS-induced apoptosis signalling cascade. Improvements over our previous modeling work on apoptosis include increased efficiency for storing and sorting waiting times of reactions, a nondeterministic approach for handling reactions competing over limited reactants and improvements, and refinements of the model reactions.
The modular nature of our systems provides flexibility with respect to future discoveries on the signal cascade. We provide a breakdown of our algorithms and explanations on improvements we have implemented. We also give an exhaustive comparison to an established ordinary differential equations technique. Based on the results of our simulations, we conclude that Membrane Systems are a useful simulation tool in Systems Biology that could provide new insight into the subcellular processes, and provide also the argument that Membrane Systems may outperform ordinary differential equation simulations when simulating cascades of reactions (as they are observed in cells).

1 Introduction

Many diseases and disorders are linked to the anomalous behavior of the apoptotic pathway within an organism. Specifically, understanding of FAS-induced apoptosis should be useful in combating cancers and HIV as pointed out in [10] and [11]. As new information is discovered on the stoichiometry and biochemical interactions involved in this pathway, scientists look to the computer simulations as a tool for understanding and predicting the effects of changes in molecules important to the pathway. In this paper, we describe two techniques based on discrete methods for modeling molecular signaling cascades within a cell. While the two discrete simulation strategies can be applied to any pathway in the cell, we implemented the new simulation techniques and chose to provide the simulation results for the caspase dependent apoptotic pathway. We provide the algorithms for our two refined approaches, a *Nondeterministic Waiting Time* algorithm and a *Multi-compartmental*

† To whom correspondence should be addressed. E-mail: apaun@latech.edu

Gillespie algorithm. As mentioned above, we have tested our methods on the FAS-induced apoptotic signal cascade and offer comparisons with a modeling technique based on ordinary differential equations as was used for the apoptotic pathway in [8].

The Nondeterministic Waiting Time (NWT) algorithm is a modified version of our Membrane System described in [2]. We provide two major improvements over our previous algorithm. First, we have added a nondeterministic logic to handle reactions competing over limited reactants. And second, we improved the runtime of the simulation algorithm by implementing a heap as a data structure used for the ordering of the reactions.

We argue that our nondeterministic, discrete approach has an advantage over methods based on ordinary differential equations, such as the ODE method found in [8]. Ordinary differential equations are an appropriate modeling technique when one seeks the average behavior of a system involving large numbers of elements – for example, modeling the dynamics of large cell populations. However, when modeling signal cascades within a single cell, we believe that differential equations may not be the best approach. We feel that a discrete method is better equipped to simulate processes that involve relatively low numbers of molecules/objects.

For example, assume that two reactions are competing over a single molecule. A differential equations technique allows both reactions to be applied, borrowing a fraction of the molecule to satisfy each equation. However, our discrete method maintains molecular integrity. In other words we do not allow a single molecule to be partially used in two reactions. The molecule is used for one reaction or the other. Our nondeterministic logic can affect the entire evolution of the system, changing the results of the signaling cascade in a way ODEs do not.

The second improvement over [2] is the use of a min-heap to store reactions. Initially, we create the min-heap in the standard (bottom-up) way. However, during the simulation our heap obeys two properties, which has led us to develop a non-standard approach to maintaining the min-heap property. The properties are the following:

(i) once intialized, the heap does not grow or shrink;
(ii) multiple nodes throughout the heap will be updated simultaneously.

In [3] the authors provide a nonstandard method for updating a min-heap. However, the min-heap they maintain does not obey (ii), so we have developed our own special maintenance functions. We provide a full description of our heap maintenance functions in a later section.

In this paper we demonstrate the design of our Nondeterministic Waiting Time algorithm, as well as our results from simulating the FAS-induced apoptotic signaling cascade. Section 2 provides a full description of our algorithm and pseudocode for our heap maintenance functions. Section 3 presents results from our simulation of the FAS-induced apoptotic pathway, along with comparisons to an established ordinary differential equations method and some experimental data. Section 4 describes a stochastic method, based on the ideas of the well-known Gillespie Algorithm. Section 5 is a discussion section providing ideas for future work.

We note that we will use the words rule and reaction interchangeably throughout this paper. In the context of Membrane Systems, we have rules associated to symbols, and in the context of cells we have reactions associated with proteins/molecules.

2 Membrane System For Simulating Signal Cascade

In the following we will give a brief description of the simulation technique proposed and then we will provide the actual algorithm with a discussion on its main ideas and an example for better understanding.

2.1 Explanation of Discrete Method

Our Membrane System follows the evolution of molecular multiplicities over time. We simulate individual chemical reactions which are asynchronous and occur discretely over different lengths of time. The rules of the our system obey the *Law of Mass Action*: the reaction rate depends proportionally on the concentrations of the reactants.

Every reaction r has an associated reaction rate constant k_r. For each reaction r we pre-compute a kinetic constant, $const_r$: $const_r = \frac{k_r}{V^{i-1} \times N^{i-1}}$ where V is the volume of the system, N is Avogadro's constant (6.0221415×10^{23}) and i is the number of reactants involved in the reaction. Consider a general second order reaction, $r_1 : A+B \rightarrow C$. We compute the amount of time required for a single application of the reaction: $wt_{r_1} = \frac{1}{const_{r_1} * |A| * |B|}$ where $|A|$ and $|B|$ represent the number of molecules of the two reactants. Or, consider a first order reaction, $r_2 : D \rightarrow E$. A single application of of the reaction is computed: $wt_{r_2} = \frac{1}{const_{r_2} * |D|}$. We refer to this as the calculation of the 'Waiting Time' (WT) of a reaction.

2.2 Nondeterministic Waiting Time Algorithm

We now provide the pseudocode of our algorithm:

1. *Initialization:* Calculate the waiting time for every reaction in the system, and label it 'WT' (time required for reaction to occur). Store the reactions in a min-heap (sorted by 'WT'). Set simulation time to zero ($t = 0$).
2. *Select Rule:* Select the reaction with the lowest waiting time. If there is a tie, go to step 3. If not, proceed to step 4.
3. *Handle Tie:* If there are enough reactants to satisfy all reactions in the tie, implement all reactions in step 4. If there are not enough reactants to accommodate all the reactions, use the nondeterministic logic to apply as many rules as possible.
4. *Apply Rule:* Update the multiplicities of the reactants and products involved in the reaction(s) from step 2. Aggregate the simulation time ($t = t$+WT).
5. *Update Rules:* Recalculate the waiting time for any reactions involving the reactants or products of the applied reaction(s). For each such reaction compare the new waiting time with the existing waiting time and keep the smallest of the two.
6. *Heap Maintenance:* Adjust the min-heap.

7. *Check Done:* If the desired simulation time has not been reached, go back to step 2.

We initially build our min-heap using the standard bottom-up technique. Once the top node of the heap has been selected (Step 2), applied (Step 4) and had its waiting time recalculated (Step 5), we leave it at the top of the tree. Often an applied reaction's new waiting time is close to its previous WT. Therefore, the node will most likely be located near the top of the heap once the heap is resorted. After recalculating the waiting time of all reactions involving reactants or products of the applied rule (Step 5), we are ready to reestablish our min-heap (Step 6). Next, we provide our special methods for heap maintenance:

```
1: Reheap()
2:     for each applied reaction (more than one if tie exists)
3:         for each product of the applied reaction
4:             for each reaction the product is a reactant of
5:                 CheckUp(reaction);
6:         for each reactant of the applied reaction
7:             for each reaction the reactant is a reactant of
8:                 CheckDown(reaction)
```

```
1: CheckUp(node r)
2:     If parent(r) exists
3:         If parent(r) > r
4:             Swap(r, parent(r))
5:             while parent(r) > children(parent(r))
6:                 Swap(parent(r), children(parent(r)))
7:             CheckUp(r)
```

```
1: CheckDown(node r)
2:     If children(r) exist(s)
3:         If r > children(r)
4:             Swap(smallestchild(r), r)
5:             while parent(smallestchild(r)) > smallestchild(r))
6:                 Swap(parent(smallestchild(r)), smallestchild(r))
7:             CheckDown(r)
```

Let us consider an applied rule of the form: A+B→C+D. When the rule is applied the system loses a molecule of A and one of B, and it also gains a molecule of C and one of D. Therefore, any reaction that has A, B, C, or D as a reactant requires a recalculation of its waiting time. After recalculation, the Reheap method is called, and the position of every reaction with a recalculated waiting time is checked. Since ties are usually infrequent, the **for** loop on line 2 of the Reheap method runs for very few iterations. Also, most reactions have at most two products and at most two reactants. Hence, the **for** loops on line 3 and 4 combine for a total of at most four iterations.

Our CheckUp and CheckDown methods differ from previous nonstandard heap maintenance [3]. Since we update multiple nodes in our heap simultaneously, we can have many nodes in violation of the min-heap property. We refer to Fig. 1 for an

example of our heap maintenance. In Fig. 1 we see three nodes in a heap of arbitrary size. Fig. 1(I) shows a clear violation of the min-heap property. Assume that the yellow and blue nodes have both just been recalculated (and their waiting time decreased), and also assume that green has not been recalculated. Our algorithm does not waste any clock cycles sorting the reactions with changed waiting times, so we can assume that CheckUp is first called on the blue node. Blue is compared with its parent (yellow), and since yellow has a smaller WT than blue, no changes to the heap are made and the CheckUp method for blue terminates.

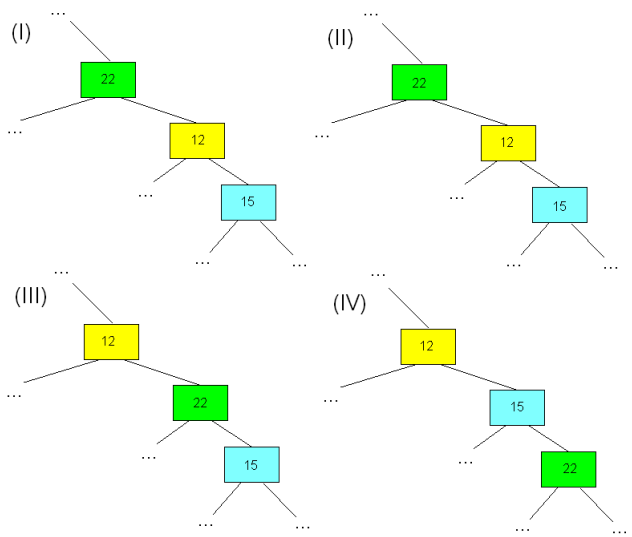

Fig. 1. (I) The waiting times for yellow and blue are recalculated, and the heap is now unsorted. (II) The method CheckUp is called for blue, but the node does not move up, since yellow has a smaller WT. (III) The method CheckUp is called for yellow. Yellow switches places with green. Before yellow checks up again, green must attempt to move down. (IV) The green node and swaps with the blue node, which satisfies the min-heap property. Green will move down as far as it can. Afterwards, CheckUp is called on the yellow node, and the process repeats until the heap is resorted.

Next, CheckUp is called for the yellow node. Yellow and green are compared, and because they violate the min-heap property, the two nodes are swapped. Since yellow and green have swapped places, the green node must be moved down the tree as far as possible in accordance with the min-heap property. Green and its children (one of which is blue) are compared, and since there is a violation of the min-heap property, green is swapped with its smallest child (assumed to be blue, but it could be the other child). Green is moved down as far as necessary, bringing up the 'tail' of yellow. Once green is in a position that does not violate the min-heap property, CheckUp is called again for yellow (line 7 of CheckUp method), and the process is repeated until yellow fails to move up. Once CheckUp and CheckDown have been called for all reactions with recalculated waiting times, the heap will be sorted.

The implementation of our heap yielded a massive performance increase over the previous algorithm from [2]. With the algorithm described in [2] we were able to complete the FAS-induced simulation ($\sim$8 cell simulated hours) in 30-40 minutes de-

pending on the particular rule set and molecular multiplicities. Our new algorithm, utilizing the efficient heap structure, takes 3-4 minutes to run the same simulations. While incorporating the heap structure we not only increased the sorting performance, we were able to eliminate an extraneous **for** loop used to put waiting times in the context of simulation times (a loop for every reaction for each applied rule). Our previous simulator had a runtime of $O(n^2 \log n)$. To be able to give the complexity of the algorithm proposed and show that in this specific case is efficient, we need to make several assumptions which are (usually) valid for the signalling cascades:
a) each reaction involves at most 5 different species of molecules
b) there are a bounded number of reactions having the same reactant (usually 3, at most 5)
c) there are not many reactions happening at the same time (due to the differences in molecule multiplicities and reaction rates).
From a), b) and c) we can now show that our new implementation has a runtime of $O(n \log n)$ with respect to the number of reactions simulated.

Next, we will describe the use of our technique in modeling the FAS-mediated signaling cascade.

3 FAS-induced apoptosis

We have chosen to simulate FAS-induced apoptosis because it has one of the most detailed descriptions/characterization in the literature (due in large part to its role in cancer and HIV research). In the interest of comparing our Membrane System with an established ordinary differential equations (ODE) technique, we have implemented 101 different rules working on 53 distinct proteins and protein complexes. The pathway begins with the stimulation of FASL and ends with the activation of the effector Caspase-3. Fei Hua et al., in [8], provide the results of an ODE simulation, as well as some experimental data (from the Jurkat cell line), which they used to fit their model.

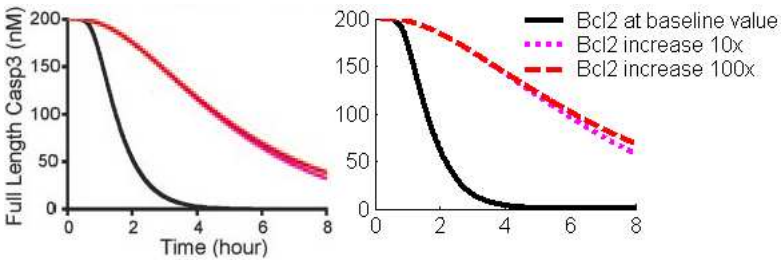

Fig. 2. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]). The two graphs show the decline over time of full length Caspase-3, which means there is an increase in active Caspase-3. In both simulators, we see that Caspase-3 nears zero after four hours for the baseline Bcl-2 concentration. However, apoptosis is inhibited as Bcl-2 levels are increased by 10- or 100-fold.

3.1 Results of Discrete Method

Similar to [8], we ran our simulation with three different initial concentrations of Bcl-2: the baseline value (75nMs), an increase by 10x (750nMs), and an increase by 100x (7500nMs). Assuming a cell volume of 10^{-12} liters, we convert the concentrations into molecular multiplicities: baseline value (45166 molecules), 10x (451660 molecules), and 100x (4516606 molecules). We expect to see a decline in Caspase-3 activation as Bcl-2 concentration is increased by 10x and 100x; we provide the results of our simulation in (Fig. 2). We also provide the results of a simulation with a decrease of 10x and 100x in comparison to the baseline Bcl-2 multiplicity (Fig. 3). Notice, the graphs based on our Membrane System simulations are comparable to the ODE results from [8].

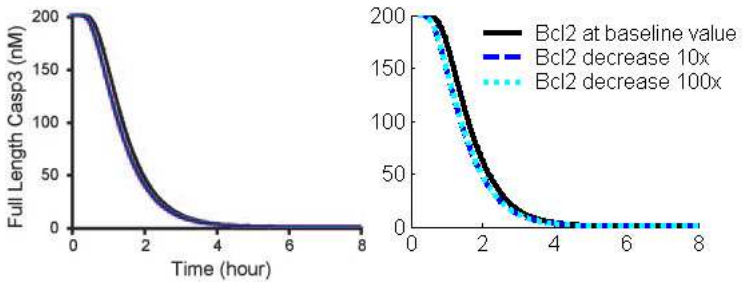

Fig. 3. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]). These results illustrate the models insensitivity to decreasing Bcl-2 concentrations by 10- and 100-fold. We see agreement between the two different simulators.

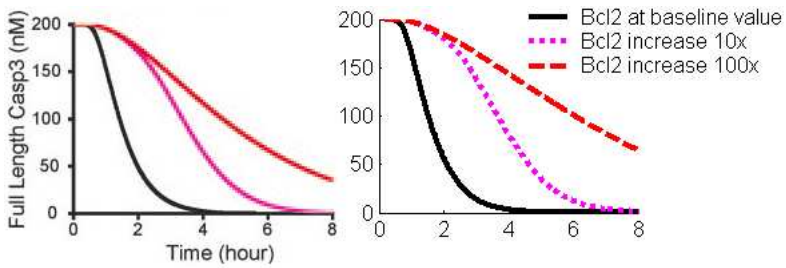

Fig. 4. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]). Here we see the effects of Bcl-2 binding to Bax only. It seems that Bcl-2 binding only with Bax is the second most effective method for blocking the apoptotic pathway.

3.2 Bcl-2's effects on the Type II pathway

We now analyze the Caspase-3 activation kinetics by considering different mechanisms through which Bcl-2 can block the type II pathway. In [24], [14], or [1] the

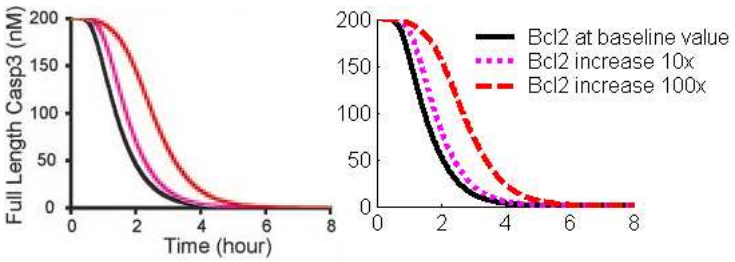

Fig. 5. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]). These two graphs show Bcl-2 binding to tBid only. We see considerably less inhibition of the pathway in this mechanism. The release of Cytochrome c is contingent on the binding of one tBid to two Bax molecules, and thus, blocking Bax would be more effective than blocking tBid.

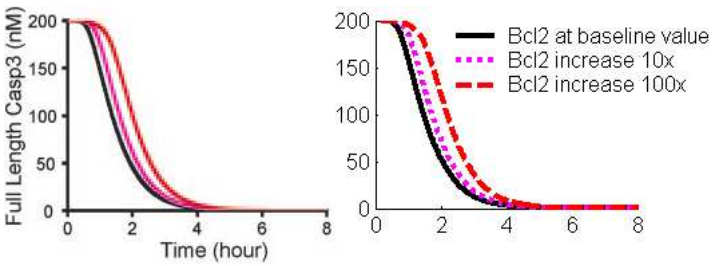

Fig. 6. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]). We see very little change in apoptotic behavior when Bcl2 is allowed to bind only with Bid. This is because Bcl-2 binding to Bid is a reversible reaction, and once Bid is truncated to tBid, it is no longer available to bind with Bcl-2.

authors suggest that Bcl-2 might bind with (a) Bax, (b) Bid, (c) tBid, or (d) bind to both Bax and tBid to block the mitochondrial pathway. We have implemented in our Membrane System four different sets of rules to test each Bcl-2 binding mechanisms. We refer the interested reader to [8] for the details of the rules. The dynamics of Caspase-3 activation are studied by varying the Bcl-2 concentration 10x and 100x the baseline value. The conclusion of [8] is that Bcl-2 binding to both Bax and tBid (d) is the most efficient mechanism for inhibiting apoptosis. Our Membrane System yields results that are in agreement with the observations from [8]. The results of (d) are illustrated in Fig. 2, and (a) - (c) can be seen in Fig. 4 - Fig. 6. A comparison of (a) - (d) at baseline Bcl-2 concentration is shown in Fig. 7.

Modeling the Behavior of the Type I Pathway

There are cells which are not sensitive to Bcl-2 over expression as described in [20]. In these cells, Caspase-3 is activated through the type I pathway, bypassing the role of the mitochondria and Bcl-2. Scaffidi et al. have suggested in [20] that the type of pathway is chosen based on the concentration of Caspase-8 generated in active form following FASL binding. High concentration of active Caspase-8 allows for direct activation of Caspase-3 (type I), but if the concentration of Caspase-8 is sufficiently low, amplification of the death signal through the mitochondria is required to induce

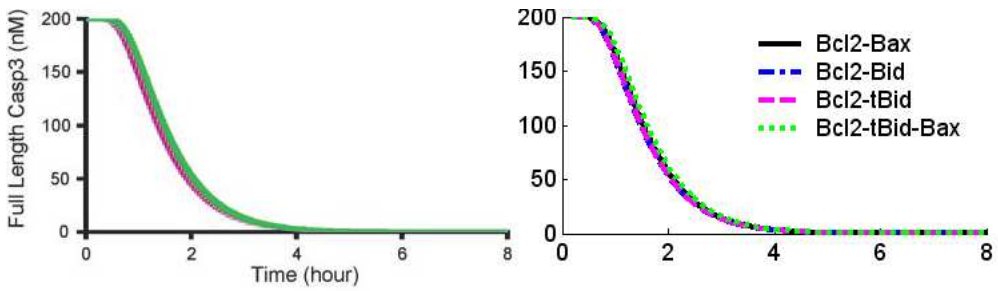

Fig. 7. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]). At baseline Bcl-2 concentration, each of the four mechanisms for Bcl-2 inhibition are similar. Both simulators agree across all rule versions: Bcl-2 binding with Bax only, Bid only, tBid only, or Bax and tBid.

the cell death (type II). We test this hypothesis by increasing the initial concentration of Caspase-8 by 20x (from 33.33nMs to 666.6nMs), which should lead to increased active Caspase-8 throughout the simulation run. In two different runs of increased Caspase-8 concentration, the baseline concentration of Bcl-2 is used and an increase of 100x, testing the sensitivity of Caspase-3 activation to Bcl-2. Fig. 8 shows the Caspase-3 activation is not sensitive to the increase in Bcl-2 concentration, which is the hallmark for type I pathway dominant behavior. N.B., the binding mechanism chosen for this simulation is Bcl-2 to Bax and tBid, which was shown above to be the most efficient mechanism for Bcl-2 inhibition of apoptosis.

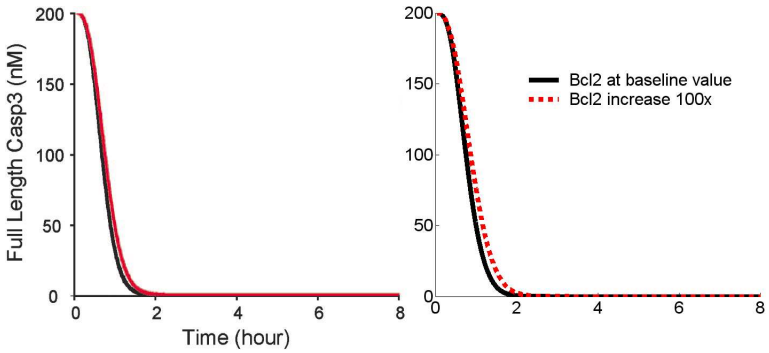

Fig. 8. On the right is the Membrane System and on the left is the ODE simulation (used with permission [8]. Unlike the type I pathway, the type II pathway is unaffected by a 100-fold Bcl-2 increase. We are pleased with these results, as Bcl-2 acts to block the release of Cytochrome c, which is an unnecessary molecule in this pathway.

We will now describe another method of simulating signal cascades, using a strategy that is based on the well known Gillespie's algorithm, but running on more than one compartment. It is called *Multi-compartmental Gillespie Algorithm.*

4 Multi-Compartmental Gillespie Algorithm

Gillespie's algorithm [4] (see also [6, 7] for some recent improvements) provides an exact method for the stochastic simulation of systems of bio-chemical reactions; the validity of the method is rigorously proved and it has already been successfully used to simulate various biochemical processes [13]. Moreover the Gillespie algorithm is used in the implementation of stochastic π-calculus [16, 22], and in its application to the modeling of biological systems [17]. The extension of the classical Gillespie algorithm, called the *Multi-compartmental Gillespie Algorithm*, is first described in [18]. Below we provide the general definition of the Membrane System.

Let $\Pi = (O, Lab, \mu, M_1, M_2, \ldots, M_n, R_1, \ldots, R_n)$ be a Membrane System with the membranes $M_i = (w_i, L_i)$ and the programs R_i, $1 \leq i \leq n$. Each set R_i of programs are active inside their corresponding membrane i. These sets contain elements of the form $(j, \pi_j, r_j, p_j, k_j)$ where:

- j is the index of a program from R_i;
- π_j is the predicate; in this section this will be always true and will be omitted;
- r_j is the boundary rule contained in the program j;
- p_j is the probability of the rule contained in the program j to be applied in the next step of evolution; this probability is computed by multiplying a stochastic constant k_j, specifically associated with program j, by the number of possible combinations of the objects present on the left side of the rules with respect to the multiset w_i (or the multiset $w_{i'}$, with $i' = upper(\mu, i)$) - the current content of membrane i (i').

First, each membrane i will be considered to be a compartment enclosing a volume, therefore the index of the next program to be used inside membrane i and its waiting time will be computed using the classical Gillespie algorithm which we recall below:

1. calculate $a_0 = \sum p_j$, for all $(j, r_j, p_j, k_j) \in R_i$;
2. generate two random numbers r_1 and r_2 uniformly distributed over the unit interval $(0, 1)$;
3. calculate the waiting time for the next reaction as $\tau_i = \frac{1}{a_0} ln(\frac{1}{r_1})$;
4. take the index j, of the program such that $\sum_{k=1}^{j-1} p_k < r_2 a_0 \leq \sum_{k=1}^{j} p_k$;
5. return the triple (τ_i, j, i).

Notice that the larger the stochastic constant of a rule and the number of occurrences of the objects placed on the left side of the rule inside a membrane are, the greater the chance that a given rule will be applied in the next step of the simulation. There is no constant time-step in the simulation. The time-step is determined in every iteration and it takes different values depending on the configuration of the system.

Next, the *Multi-compartmental Gillespie's Algorithm* is described in detail:

1. *Initialization:*

- set time of the simulation $t = 0$;
- for each membrane i in μ compute a triple (τ_i, j, i) by using the procedure described above; construct a list containing all such triples;
- sort the list of triples (τ_i, j, i) according to τ_i;

2. *Iteration:*
   - extract the first triple, (τ_m, j, m) from the list;
   - set time of the simulation $t = t + \tau_m$;
   - update the waiting time for the rest of the triples in the list by subtracting τ_m;
   - apply the rule contained in the program j only once changing the number of objects in the membranes affected by the application of the rule;
   - for each membrane m' affected by the application of the rule remove the corresponding triple $(\tau'_{m'}, j', m')$ from the list;
   - for each membrane m' affected by the application of the rule j re-run the Gillespie algorithm for the new context in m' to obtain $(\tau''_{m'}, j'', m')$, the next program j'', to be used inside membrane m' and its waiting time $\tau''_{m'}$;
   - add the new triples $(\tau''_{m'}, j'', m')$ in the list and sort this list according to each waiting time and iterate the process.
3. *Termination:*
   - Terminate simulation when time of the simulation t reaches or exceeds a preset maximal time of simulation.

Therefore, in this approach, the waiting time computed by the Gillespie algorithm is used to select the membranes which are allowed to evolve in the next step of computation. Specifically, in each step, the membranes associated to programs with the same minimal waiting time are selected to evolve by means of the corresponding rules. Moreover, since the application of a rule can affect more than one membrane at the same time (e.g., some objects may be moved from one place to another), we need to reconsider a new program for each one of these membranes by taking into account the new distribution of objects inside them. Note that in this point our approach differs from [21] where only one program is applied at each step without taking into account the rest of the programs that are waiting to be applied in the other membranes, neither it is considered the disruption that the application of one program can produce in various membranes.

We coded the model in C and simulated the FAS-induced apoptotic pathway described in Section 3. In the interest of saving space, we provide only some of the stochastic simulation results in our discussion section.

5 Discussion And Final Remarks

We have provided two discrete methods for modeling molecular signaling cascades, highlighting key changes to our previous technique. We also gave the results from the simulation of the FAS-mediated apoptotic pathway. Our Membrane System has yielded comparable results to an ordinary differential equations technique. The sixteen distinct simulations reach apoptosis at similar rates to the ODE method (as shown in Fig. 2 through Fig. 8). Although the activation of Caspase-3 is similar

between the two techniques, the molecular interactions throughout are different. We have compared the results of our two simulators with the experimental results obtained by Fei Hua et al. The Caspase-3 results are not very surprising, but the activation of Caspase-8 raises our interest. Refer to Fig. 9 for a comparison of the three simulation methods. We note the fact that in the case of the stocastic system we have depicted a single run of the simulation.

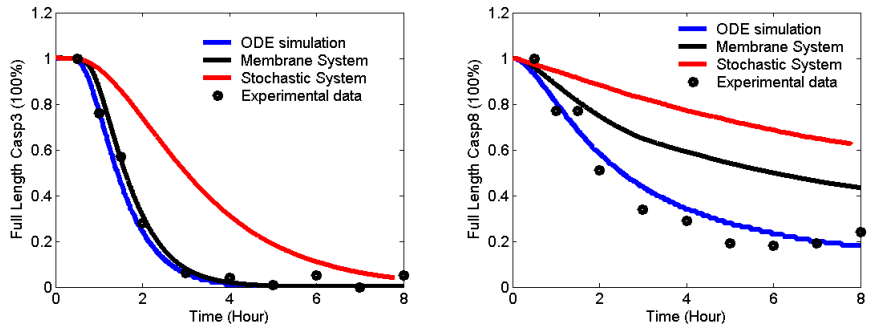

Fig. 9. The experimental data and ODE simulation results were obtained from Fei Hua et al. to be used in comparison to our Membrane System simulator. In both simulators, Bcl-2 is binding with Bax and tBid. We see that the decrease of full length Caspase-3 is similar in all three results. Interestingly, the decline of full length Caspase-8 is least prominent in the Membrane System simulator. The contrast could be a result of the discrete nature of our system. As for both results being different than the experimental data, we believe that further investigation of kinetic rates of the reactions will allow for better agreement between simulation and experimentation.

The consistency between the framework and the experimental results in the paper [8] validates our model. We have stated that our discrete methods handle low levels of molecules in a different way that ODE techniques. To further investigate the differences between discrete and ODE methods, we have chosen to focus on one rule from the FAS-mediated pathway (a transformation): $[\, CASP8_2^{P41} \,]_c \rightarrow [\, CASP8_2^{*} \,]_c$. The initial concentration of $CASP8_2^{P41}$ is assumed to be $.03nMs$ (or 18 molecules) and we use the same kinetic rate $k_5 = 0.1s^{-1}$ as in the model used in the current paper.

The ode45 method in MATLAB was used to obtain the results for the ODE technique, and is shown as the blue line in the Fig. 10. The membrane system was used to produce the discrete results depicted in the dark blue bars. Fig. 10, clearly shows a divergence between the two techniques. At the end of the simulation (second 16) we notice that the ODE has the value approximately 14 whereas the membrane simulator has the value 18. These results are obtained for the exact same constants and reaction, leading to a difference greater than 20% of the end value. We suggest to the interested reader to contemplate what would be the effects of similar differences in protein multiplicities/concentrations when considering hundreds or thousands of rules in the model being simulated. It should be obvious now why the discrete simulation techniques are producing different results than the continuous simulations. We believe that when modeling signaling cascades over a relatively

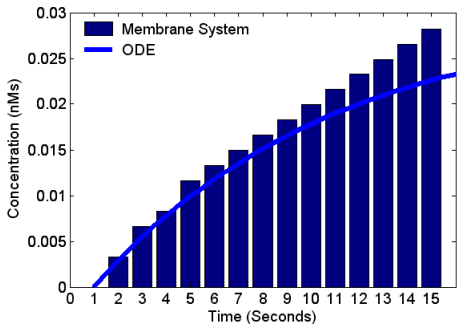

Fig. 10. The Membrane System does not allow the existence of fractional molecules, which makes it fundamentally similar to the real world. Since ODEs use concentrations for simulation, the data can become skewed as the simulation commences - i.e., fractions of molecules are interacting. If the concentration is sufficiently high, this may not affect the quality of results, but as we can see here, deviations occur when dealing with evolution of multiplicities rather than concentrations.

small number of molecules, the discrete methodology may yield better/more-realistic results than an ODE technique.

Our Nondeterministic Waiting Time algorithm shows that Membrane Systems are an intriguing alternative to ordinary differential equations methods. We have argued that the discrete nature of our technique might be better for simulating the evolution of systems involving low numbers of molecules. In the future, we would like to add a stochastic element to the Nondeterministic Waiting Time algorithm, allowing for limited stochasticity in our simulations. We will also be applying our model to signal cascades other than FAS-mediated apoptosis. Another future thrust of our group will be in the extension and refinement of the pathway discussed here. We plan to model the behavior of the caspase-based apoptotic pathway in the HIV infected cells.

Acknowledgements

J. Jack gratefully acknowledges a Ph.D. fellowship from the University of Louisiana System and support from NSF Grant CCF-0523572. The research of O. H. Ibarra was supported in part by NSF Grants NSF Grants CCF-0430945 and CCF-0524136. The research of A. Păun was supported in part by LA BoR RSC grant LEQSF (2004-07)-RD-A-23 and NSF Grants IMR-0414903 and CCF-0523572.

References

1. Cheng, E.H., Wei, M.C., Weiler, S., Flavell, R.A., Mak, T.W., Lindsten, T., Korsmeyer, S.J. (2001). BCL-2, BCL-XL sequester BH3 domain-only molecules preventing BAX- and BAK-mediated mitochondrial apoptosis. *Molecular Cell*, **8**, 705–711.
2. Cheruku, S., Păun, A., Romero-Campero, F., Pérez-Jiménez, M., Ibarra, O. (2006). Simulating FAS-Induced Apoptosis by Using P Systems. *Proceedings of Bio-inspired computing: theory and applications (BIC-TA)* September 18-22, 2006, Wuhan, China, also extended version accepted to *Progress in Natural Science*, **17**(4), 424–431.

3. Gibson, M. A., Bruck, J. (2000). Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *J.* Phys. Chem. **A 104**, 1876–1889.
4. Gillespie, D.T. (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics*, **22**, 403–434.
5. Gillespie, D.T. (1977). Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, **81**, 25, 2340–2361.
6. Gillespie, D.T. (2001). Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems. *Journal of Chemical Physics*, **115**, 4, 1716–1733.
7. Gillespie, D.T. (2003). Improved Leap-size Selection for Accelerated Stochastic Simulation. *Journal of Chemical Physics*, **119**, 16, 8229–8234.
8. Hua, F., Cornejo, M., Cardone, M., Stokes, C., Lauffenburger, D. (2005). Effects of Bcl-2 Levels on FAS Signaling-Induced Caspase-3 Activation: Molecular Genetic Tests of Computational Model Predictions. *The Journal of Immunology*, **175**, 2, 985–995 and correction **175**, 9, 6235–6237.
9. Ibarra, O.H., Păun, A. (2005). Counting time in computing with cells. *Proceedings of DNA Based Computing, DNA11*, London, Ontario, 25–36 and *Lecture Notes in Computer Science*, **3892**, (2006), 112–128.
10. Krammer, P.H. (2000). CD95's deadly mission in the immune system. *Nature*, **407**, 789–795.
11. Krammer, P.H., (2000). CD95's deadly mission the in immune system. *Nature*, **407**, 789–795.
12. Manca, V., Bianco, L., Fontana, F. (2005). Evolution and Oscillation in P Systems: Applications to Biological Phenomena, *Lecture Notes in Computer Science*, **3365**, 63 – 84.
13. Meng, T.C., Somani S., Dhar, P. (2004). Modelling and Simulation of Biological Systems with Stochasticity. *In Silico Biology*, **4**, 3, 293–309.
14. Oltavi, Z.N., Milliman, C.L., Korsmeyer, S.J. (1993). Bcl-2 heterodimerizes in vivo with a conserved homolog, Bax, that accelerates programmed cell death. *Cell*, **74**, 4, 609–619.
15. Păun A., Pérez-Jiménez M., Romero-Campero F. (2006). Modelling Signal Transduction using P Systems, *Lecture Notes in Computer Science*, **4361**, 100–122.
16. Philips, A., Cardelli. L. (2004). A Correct Abstract Machine for the Stochastic Pi-calculus. *Proc. Bioconcur04*. ENTCS.
17. Priami, C., Regev, A., Shapiro, E., Silverman, W. (2001). Application of a Stochastic Name-Passing Calculus to Representation and Simulation of Molecular Processes. *Information Processing Letters*, **80**, 25–31.
18. Pérez-Jiménez, M.J., Romero-Campero, F.J. (2006) P Systems, a New Computational Modeling Tool for Systems Biology, *Transactions on Computational Systems Biology*, **4220**, 176–197.
19. Romero-Campero, F.J., Pérez-Jiménez, M.J. (2005). A Study of the Robustness of the EGFR Signalling Cascade using Continuous Membrane Systems. *Lecture Notes in Computer Science*, **3561**, 268 – 278.
20. Scaffidi, C., Fulda. S., Srinivasan, A., Friesen, C., Li, F., Tomaselli, K.J., Debatin, K.M., Krammer, P.H., Peter, M.E. (1998). Two CD95 (APO-1/Fas) signaling pathways. *The Embo Journal*, **17**, 1675–1687.
21. Stundzia, A.B., Lumsden, C.J. (1996). Stochastic Simulation of Coupled Reaction-Diffusion Processes. *Journal of Computational Physics*, **127**, 196–207.
22. The Stochastic Pi-Machine: `http://www.doc.ic.ac.uk/~anp/spim/`.
23. Van Kampen, N.G. (1992) Stochastic Processes in Physics and Chemistry. Elsevier Science B. V., Amsterdam, The Netherlands.
24. Wang, K., Yin, X.M., Chao, D.T., Milliman, C.L., Korsmeyer, S.J. (1996). BID: a novel BH3 domain-only death agonist. *Genes & Development*, **10**, 2859–2869.