Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

IMPLEMENTATION OF A DISCRETE EVENT SIMULATOR FOR BIOLOGICAL SELF-ASSEMBLY SYSTEMS

Tiequan Zhang
Rori Rohlfs
Russell Schwartz

Dept. of Biological Sciences and Computer Science Dept.
Carnegie Mellon University
4400 Fifth Avenue
Pittsburgh, PA 15213, U.S.A

ABSTRACT

We have implemented a simulation tool for the study of computationally challenging biological self-assembly systems, particularly viral protein shells. The simulator implements a generic model of self-assembly based on simple local binding interactions to specify the behavior of complex self-assembly reactions. Recently developed discrete event methods allow for fast quantitative simulation of these systems. The new simulator uses the Java language to implement the model in a portable, interactive graphical tool. The Java libraries can also be used directly to build customized simulations. This paper discusses the simulator model, the theoretical basis for its efficient operation, and implementation issues in its design. It also discusses empirical validation of the simulator package and presents sample applications.

1 INTRODUCTION

Biological self-assembly is a process by which molecules, typically repeated proteins subunits, spontaneously form into larger, complex structures with limited help from cellular machinery. Typical biological self-assembly systems include actin and tubulin filaments, which form a network of protein filaments called the cytoskeleton that is critical to numerous processes in eukaryotic cells, and virus capsids, which form the protective outer coat around the nucleic acid of a virus. Investigations of these self-assembly systems are important for basic biology and medical research. In addition, they hold important lessons for humans attempting to design novel self-assembly systems (Whitesides 1995, Whitesides 2002). However, limitations in our ability to observe and manipulate rapid, nanometer scale assembly reactions make it difficult to gain an understanding of these systems by traditional experimental approaches alone. These problems are particularly acute for viral capsids, which are typically extremely complex and robust systems of hundreds of proteins assembled into icosahedrally symmetric structures. Simulation methods provide a means to address some of these difficulties and assist in the planning and interpretation of experimental work.

Virus capsid assembly has recently attracted particular attention in the modeling community because of its complexity, its efficiency, and the difficulties involved in experimentally observing its progress. Current understanding of virus capsid assembly is based on the Caspar and Klug theory of "quasi-equivalence" (Caspar and Klug 1962, Caspar 1980), which offered a possible explanation for observed shell symmetries. Quasiequivalence theory also provided a taxonomy for observed shell structures in terms of "T numbers" describing the relative positions of subunits in the shell. Other approaches and models have been developed to investigate different aspects of virus capsid self-assembly, such as the equilibrium behavior of assembly systems (Bruinsma et al. 2004, Zlotnick 1994), the favored assembly pathways (Reddy et al. 1998), mechanisms behind some unusual "non-quasiequivalent" structures (Schwartz, Garcea, and Berger 2000; Twarock 2004), and the overall reaction kinetics of the assembly process (Zlotnick 1994, Endres and Zlotnick 2002).

Several simulation models have been developed based on a theory of virus assembly called local rules (Berger, et al. 1994). Local rules theory proposed that virus capsid formation can be directed by local interaction of virus coat protein subunits, with complex capsid geometries arising from subunits selecting their local geometries and binding partners based only on their immediate environments in the shell. The simplest local rules set, used to describe a T=1 virus capsid geometry, is illustrated in Figure 1. This model was used to explain the normal geometrical structures of virus and the possible reasons for some malformed assemblies. It later formed the basis for a more complex simulation model called local rules dynamics that combined local rules with a sophisticated molecular dynamics-

like model of particle movement and structure and reaction rates (Schwartz et al. 1998, Schwartz 2000). However, the computational cost of this model made it difficult to simulate large-scale assembly and systems with events acting on widely different time-scales.

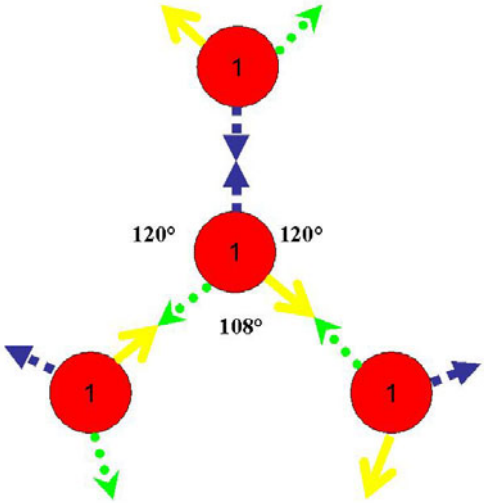

Figure 1: Local Rules for T=1 Virus Capsid Geometry

A novel discrete event method for accelerated quantitative simulation of self-assembly reaction progress was proposed by Jamalyaria, Rohlfs and Schwartz (2005). This fast, memory-efficient simulation method was designed to handle complex self-assembly systems on the scales of single cells. Virus capsids are challenging in general because the enormous number of distinct intermediates possible makes it computationally intractable to use differential equation approaches commonly used to approximate chemical reactions on large scales without large simulations. The cellular scale is particularly challenging because it encompasses system sizes that are near the boundary of what is computational achievable by stochastic discrete event methods but are sufficiently small that deterministic continuous methods cannot necessarily be considered reliable approximations.

In the present work, we describe a simulator developed by combining local rules theory and our fast discrete-event simulation method to model these hard systems. The objective of this implementation is to build a reliable research and educational tool to investigate the pathways and possible kinetic traps in self-assembly systems; develop and evaluate hypothesis about actual systems with the aid of available experimental data; facilitate a thorough analysis of the trade-offs among run time, memory usage and accuracy; and analyze the convergence of the discrete and continuous methods for systems scales near the boundary of what is accurately approximable by large-scale methods. It is specifically designed to:

- Employ a very general model adaptable to many different biological geometries and assembly processes.
- Explicitly represent the stochastic nature of biochemistry on small (cellular) scales.
- Achieve accuracy and robustness for a broad range of system parameters and model details.
- Be sufficiently portable and easy to use that it can be productively applied by users with no familiarity with the theory of the simulation methods.

In the remainder of this paper, we describe the simulator model and implementation and present empirical results demonstrating the simulator capabilities and validating its correctness.

2 IMPLEMENTATION

2.1 Overview

This simulator has been developed using a Model-View-Controller (MVC) design pattern. The model represents the local rules that govern behaviors of the assemblies, which are the basic unit of the simulation, the event queue and a set of global properties of the simulation (e.g. solution volume or temperature). The view provides an interactive display of simulation progress and renders the detailed three-dimensional graphical representation of the structures present in the simulation at any given time. The controller, which handles interactions between the user and the model and view, includes two separate components: that instructing the simulator to advance the time or change the angle of the view and that implementing the laws of physics that determine the time course of the simulation. This separation reflects the centrality of the physics model to the simulator and the conceptual distinction between predictable user-directed actions and unpredictable actions the simulator itself directs. Users can customize the important parameters of the physical model and control the simulation through either the Graphical User Interface (GUI) or through direct access to the Java code.

2.2 Discrete Event Model

Chemical reactions are conventionally described by formulae such as the following:

$$A + B \underset{k_r}{\overset{k_f}{\rightleftharpoons}} C.$$

The above states that two molecules, A and B, react to form a molecule C with a characteristic rate k_f. C can also convert back into A and B with a rate k_r. Systems of such reactions are often deterministically modeled using ordinary differential equations (ODEs) to track the concentra-

tions of reactants such as A, B and C as functions of time and of the rate constant parameters (Turner, Schnell, and Burrage, 2004). By contrast, a discrete event (DE) model uses a stochastic approach to model the reaction progress in terms of individual reaction events separated by waiting times. Waiting times are exponentially distributed with average times T_f and T_r for each type of reaction. If we set our units so that the total volume of the simulation is 1, then we can easily relate ODE and DE models of the same system. If we specify the time unit such that $T_f = 1/k_f$ and $T_r = 1/k_r$, then the discrete count of reactants in the DE model will converge on the continuous concentration of reactants in the ODE model for sufficiently large systems.

2.3 Protein Model

The model used by the present simulator is derived from local rules theory (Berger et al. 1994), a model defining self-assembly systems in terms of simple pair-wise binding interactions. Local rules theory is a hypothesis about the mechanism of actual virus shell assembly, but is also a very useful computational abstraction in that it establishes a general model allowing for the concise specification of a broad range of self-assembly systems. The physical model of the simulator is similar to the local rules dynamics model (Schwartz et al. 1998, Schwartz 2000), a prior extension of local rules to handle reaction rate information. The new model, however, uses a discrete-event simulation method (Jamalyaria, Rohlfs and Schwartz 2005), in contrast to the computationally demanding molecular dynamics-like method of the local rules dynamics model.

The basic building block of a simulation is a subunit, which is generalized to represent a single protein or capsomer in a biological self-assembly system. Each subunit has a user-specified set of binding sites defined by a position in space relative to the center of the subunit and a set of other binding sites with which it can bind. The affinities of each type of binding site with its own sets of compatible types of binding sites are encoded by the mean waiting times to form and break a binding interaction, T_a and T_d. Because waiting times are exponentially distributed, the means completely specify the waiting time distributions.

Some assembly models require that subunits can dynamically change their binding configurations. One example is the autostery model (Caspar 1980), which provides an explanation for the experimental observation that the overall rate of growth is limited by the time to form a small "growth nucleus" (Prevelige, Thomas, and King 1993). In order to model changing or partially changing binding site configurations, subunits in this simulator are defined as combinations of domains, each of which has some possible conformations. A Domain object encodes a partial current pattern of binding sites in the subunit. Using this representation, the conformations of individual domains can change without affecting other domains. Thus, some binding sites may become available or unavailable due to some stimulus while other binding sites remain unperturbed. In the simplest case, there is only one domain in a subunit, which might contain two conformations, one with some binding sites, the other without any binding site. Two such conformations are graphically illustrated in Figure 2 (a) and (b). Figure 3 illustrates the process by which one domain can switch between two conformations, a and b, with different affinities represented by average switching times T_f and T_b, while a second domain remains fixed. The sphere-cone complex shown in Figure 2(b), (c) and (d) are graphical representations of structures of some subunits. The physical properties (mass, size etc.) and the hierarchical structure of one subunit – domain, conformation and binding site – define its specific binding behaviors.

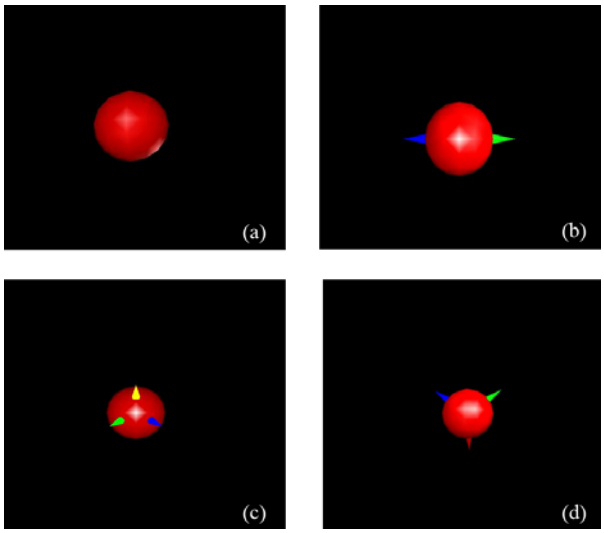

Figure 2: Subunits from Different Assembly Models

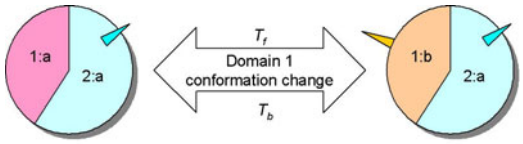

Figure 3: Conformational Switching in Domain 1

The next level in the conceptual hierarchy is an assembly. An assembly consists of either one subunit (also called a monomer) or multiple subunits connected by bonds from the binding of pairs of compatible binding sites. Some examples of possible assembly structures that can be represented by the model are shown in Figure 4. Figure 5 illustrates the hierarchical architecture used to implement an assembly in this simulator. The binding site configuration and binding patterns predefined by the local rules for a specific system determine the structure of the possible intermediates and final assembly products.

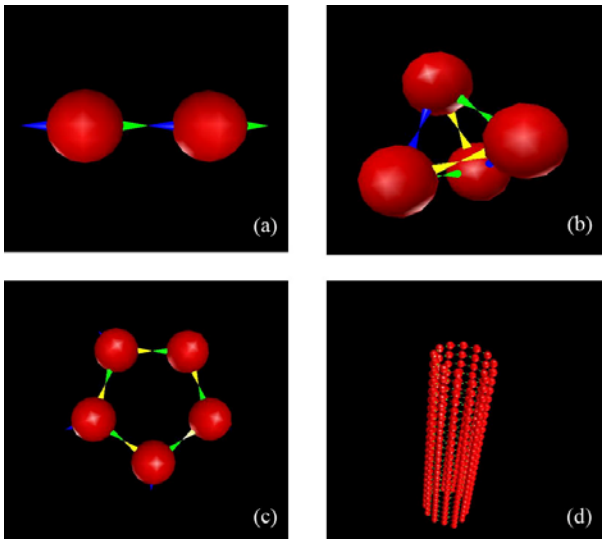

Figure 4: Examples of Assembly Structures: (a) Dimer Filament (b) Tetrahedron (c) Pentamer of Virus Capsid Model (T=1) (d) Helical Tubular Filament

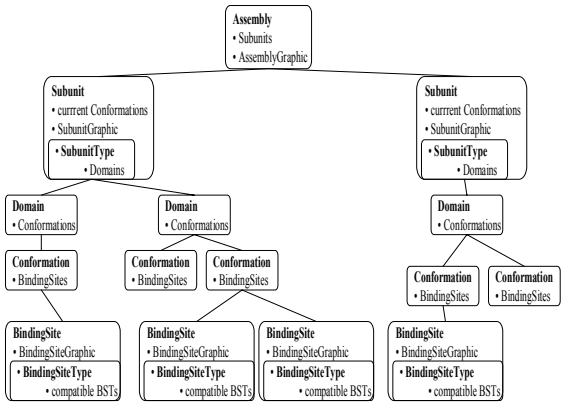

Figure 5: Hierarchical Architecture of Assembly Implementation in the Simulator

2.4 Event Implementation

A set of discrete events, representing individual chemical reactions, are used to advance time in the simulation. The simulator can explicitly represent, sample and process the following events:

1. FormBondEvent. This type of event represents a future assembly binding reaction involving two compatible binding sites on different subunits from same or different assemblies.
2. BreakBondEvent. This type of event represents a future breaking reaction of a binding interaction between two compatible binding sites. Fulfillment of this reaction will separate two previously bound binding sites.
3. ConfChangeEvent. This type of event represents a change in conformation of a subunit domain.

Those types of events are implemented as a set of Java classes, and contain such essential information as the time when the event is sampled and will be processed and the binding sites or domains it involves.

All the possible events for each Assembly object are sampled from exponential distributions based on a user-defined average waiting time for each reaction, which is equivalent to the inverse of the reaction rate constant in more standard chemical kinetics notation. For example, each bond in an assembly is sampled for possible BreakBondEvents and each free binding site in the assembly is sampled for possible FormBondEvents with each free compatible binding site in the simulation. The event with minimum waiting time found for each assembly is stored into an event queue. This model of exponentially-distributed waiting times is mathematically equivalent to the N-fold way model for reaction kinetics (Gillespie 1976), which is itself a continuous-time Markov model representation of the system. We use a slightly modified version of a more efficient algorithm than that used in the N-fold way method that is better suited to systems with large numbers of possible reaction intermediates (Jamalyaria, Rohlfs and Schwartz 2005).

The simulation stores sampled events in an event queue, which is sorted according to the time that an event is supposed to be processed. The event queue is currently implemented as a binary heap priority queue. Adapting other queue structure for some particular types of assembly simulation could improve the average performance, although more experiments are needed to find such application domains.

2.5 Simulation Process Implementation

2.5.1 Initialization

At the start-up of simulation (simulation system time t =0), the system is provided with a population of m assemblies and an empty event queue. For each assembly, we sample all possible events involving that assembly and place the one with minimum waiting time into the event queue. The sampling of FormBondEvent involves pair-wise interactions among all assemblies. To avoid duplicate FormbondEvent sampling and enqueueing, $m(m-1)/2$ comparisons are made and the produced list of events is screened to remove extra identical events. Currently, this method of initialization, which requires a one-time $O(m^2)$ computation, is the practical bottleneck for our simulation computations. We expect this situation to be improved by using a technique of the classical N-way method to equate small iden-

tical assemblies during sampling; although this technique cannot be applied to arbitrary assemblies because of the intractability of testing for identity between assemblies, it should offer a substantial reduction in startup costs if applied to monomers.

2.5.2 Simulation Run

After a minimum-time event is extracted from the queue, the current simulation time is updated. If the most recent time when all assemblies involved in this event change state is no later than that when this event is sampled, this event is a valid event, otherwise it is considered invalid. Different types of events are handled as follows:

- Valid BreakingBondEvent: When the breaking can split the involved assembly into two unconnected sub-assemblies, we sample new events for them and change their states (events to be involved, sampling time, component information etc.). Extra rules can be applied for some special cases. For example, in some linear filament simulations, it is desirable to allow binding interactions to break only at the ends of the filaments. Bonds breaking within closed loops are also sometimes treated as a special case (e.g. a bond breaking in the pentamer shown in Figure 4(c)). In this case, we can use a FormBondEvent with a shorter waiting time to rebind the broken bond quickly. This fast rebinding indirectly represents the difficulty of breaking a stable loop structure, due to the entropy benefit of binding subunits already held in the proper binding positions by other binding interactions.
- Valid FormBondEvent: Normally, for a FormBondEvent involving two assemblies, they are merged into a single assembly and new minimum-time event is sampled for that assembly. Binding can also occur between compatible binding sites within a single assembly, triggering resampling of events for that assembly. Such binding is allowed only if the binding sites are of compatible types and are within user-specified bond angle, distance, and rotation tolerances of one another. A queued binding event might actually be impossible to implement, either because the two assemblies have subunits that would overlap if they were bound (a problem called steric hindrance) or because the number of subunits is larger than a user-specified maximum assembly size. In such cases, the binding event will not be implemented and new events will be sampled for the two assemblies.
- Valid ConfChangeEvent: The current conformation of a monomer will be switched to another conformation specified in this event. This also triggers the sampling of new possible events for the assembly involved.
- Invalid event: An invalid event does not trigger any change in the simulation state. However, if one actor in the invalid event has not had new events sampled for it since the invalid event, then a new minimum time event will be sampled for that assembly. This ensures that at all times there is some sampled event for each assembly in the queue.

During the simulation run, the resampling of new events still involves obtaining one type of event with the minimum time. However, since at most two assemblies need to be assigned new events, no more than two events are sent to the queue at each step, unlike in initialization. Furthermore, the amount of work per step is linear in the system size after the initial quadratic startup cost.

Figure 6 outlines the event processing loop. After a predefined number of simulation steps are finished or the event queue becomes empty (because no more events are possible), the simulation process will stop to wait for new controls from the user.

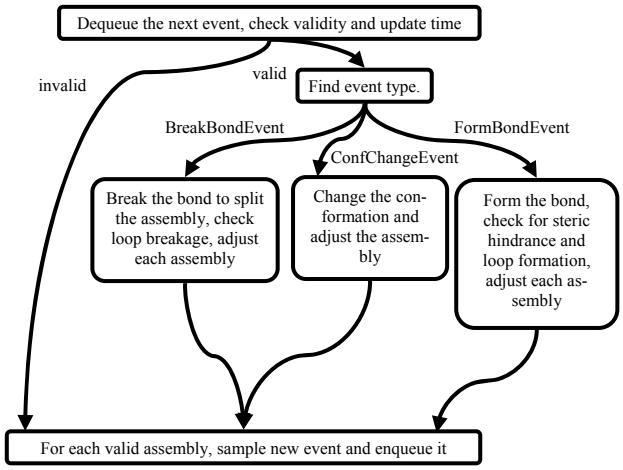

Figure 6: Flow Chart of a Simulation Step

2.6 View and Control

Users can use the mouse to rotate, translate or zoom part of the simulation scene to observe the details of the assembly structures from different viewpoints. Users can control the simulation process through the initialization of some parameter values listed in Table 1 and through GUI buttons which provide options to run the simulation continuously or step-by-step. More sophisticated control of the simulation can be obtained by editing additional parameters or subunit type definitions directly in the Java code defining a specific simulation run.

Table 1: Parameters of Simulation and Kinetics Equations

Parameter	Description
A_0	Monomers (assembly with one subunit) without binding capacity
A_1	Monomers with binding capacity
A_j	Assembly consisting of j subunits
$[A_j]$	The concentration of assemblies with j subunits
N	The initial number of monomers
l	The maximum number of subunits allowed in any filament
T_a	The average waiting time for two binding sites to associate to form a bond. It is equal to $1/k_a$ in kinetics equations.
T_d	The average waiting time for two binding sites to break a bond. It is equal to $1/k_d$ in kinetics equations.
T_+	The average waiting time for a monomer to switch from a non-binding to a binding conformation. It is equal to $1/k_+$ in kinetics equations.
T_-	The average waiting time for a monomer to switch from a binding to a non-binding conformation. It is equal to $1/k_-$ in kinetics equations.

3 EXAMPLE APPLICATIONS AND RESULTS

We validated our prototype simulator primarily by comparisons of discrete event and ODE models of simple structures. Given sufficiently large numbers of subunits, the two models would be expected to converge on identical time courses. The simulator was also used to build a simplified virus capsid model with T=1 icosahedral symmetry efficiently, a system for which we cannot construct a tractable ODE model.

Simplified virus capsid model: This simulator has been used to build a simplified virus capsid model with T=1 icosahedral symmetry. The complete assembly, shown in Figure 7, includes 60 subunits with the structure shown in Figure 2(d). The local rules set used is shown in Figure 1.

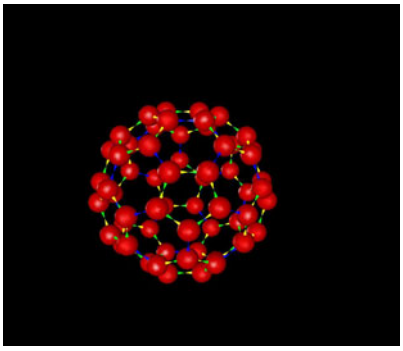

Figure 7: T=1 Icosahedral Virus Capsid Screenshot

Linear filament assembly model: The structure of the subunit of actin-like assembly is shown in Figures 2(a) and 2(b). In this type of simulation, the following rules are set:

- The maximum allowable length of a filament assembly is set to l=4.
- Each monomer is able to switch between binding and non-binding conformations.
- The simulation is initialized with N monomers in the non-binding conformation.
- The FormBondEvent has to involve at least one monomer.
- The BreakingBondEvent has to involve one subunit located at either end of a filament assembly.

After the simulation reaches equilibrium (shown in Figure 8), the average numbers of assemblies of each size over some time are calculated to analyze the distribution of assemblies.

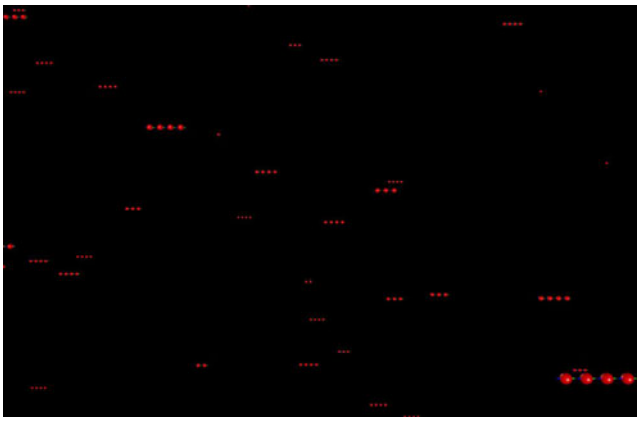

Figure 8: Screenshot of Filaments Distribution at Equilibrium after 40,000 Steps with Parameters N=500, L=4, T_a=1, T_d=0.25, T_+=0.2, T_-=1

The corresponding kinetic reaction equations are listed below:

$$
\begin{aligned}
A_0 &\underset{k_-}{\overset{k_+}{\rightleftharpoons}} A_1 \\
A_1 + A_1 &\underset{k_d}{\overset{k_a}{\rightleftharpoons}} A_2 \\
A_1 + A_2 &\underset{2k_d}{\overset{2k_a}{\rightleftharpoons}} A_3 \\
&\vdots \qquad\quad \vdots \qquad \vdots \\
A_1 + A_{l-1} &\underset{2k_d}{\overset{2k_a}{\rightleftharpoons}} A_l .
\end{aligned}
$$

A system of ordinary differential equations (listed in equations (1)) based on the kinetic reactions are set up and nu-

merically integrated using MATLAB. Thus, to verify the correct implementation of the queue-based discrete event simulator, it is relatively straightforward to compare the results from the two models, as shown in Figure 9. The results show the similar distributions both in values and patterns.

$$\begin{aligned}
\frac{d[A_0]}{dt} &= -k_+[A_0] + k_-[A_1] \\
\frac{d[A_1]}{dt} &= -2k_a \sum_{j=1}^{l-1} [A_1][A_j] - k_-[A_1] + 2k_d \sum_{j=2}^{l} [A_j] + k_+[A_0] \\
\frac{d[A_2]}{dt} &= -2k_a[A_1][A_2] - k_d[A_2] + 2k_d[A_3] + k_a[A_1]^2 \\
\frac{d[A_j]}{dt} &= 2(-k_a[A_1][A_j] - k_d[A_j] + k_d[A_{j+1}] + k_a[A_1][A_{j-1}]) \\
& \quad (2 < j < l) \\
\frac{d[A_l]}{dt} &= -2k_d[A_l] + 2k_a[A_1][A_{l-1}].
\end{aligned} \tag{1}$$

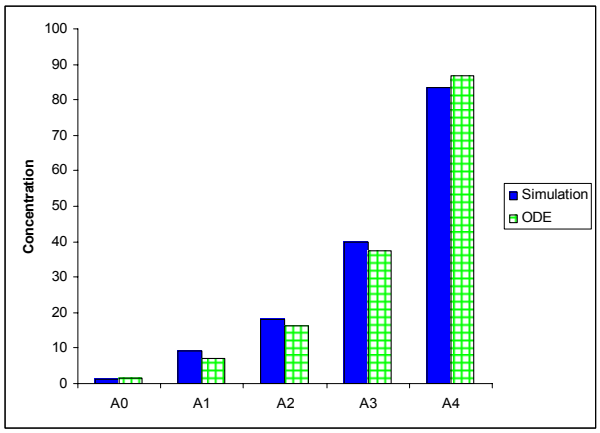

Figure 9: Comparison of Filament Length Distributions at Equilibrium

Platonic solid assembly model: Simulating virus capsid self-assembly is more challenging and complex than simulating filament assembly. Because of the large number of possible intermediates for even the simplest icosahedral structures, we cannot construct an ODE model for comparison without substantial simplification of possible assembly pathways. To verify the accuracy of the methods required for icosahedral capsid assembly simulation but not for filament simulation (e.g. loop and steric hindrance detection, fast binding following a loop breaking), we used the much simpler tetrahedron system. Although tetrahedrons are not to our knowledge biologically significant, they depend on the same loop-handling code as icosahedra, but are simple enough to allow us to construct ODE models. The following rules are set for the simulator:

- The simulation is initialized with N monomers with binding capacity.
- The bond binding and breaking can involve assemblies of any size.
- The three types of binding sites on each subunit are set to have identical waiting time distributions.
- When a bond breaking produces an open loop instead of two assemblies, the broken bond would be "sealed" instantly, effectively assuming infinite rate for binding sites held in place by other interactions.

The kinetic reaction equations for this system are shown below:

$$\begin{aligned}
A_1 + A_1 &\underset{k_d}{\overset{\frac{3}{2}k_a}{\rightleftharpoons}} A_2 \\
A_1 + A_2 &\xrightarrow{4k_a} A_3 \\
A_1 + A_3 &\xrightarrow{3k_a} A_4 \\
A_2 + A_2 &\xrightarrow{\frac{4}{3}k_a} A_4.
\end{aligned}$$

In the above reaction equations, A_1 represents a monomer with three different binding sites and A_2 represents a dimer with four free binding sites. A_3 has a triangular shape with three free binding sites. A_4 is the complete tetrahedron without any free binding site. The reaction rate constants are determined for each reaction by the number of possible pairs of binding sites that could implement that reaction and by the reaction rate constants of the individual binding site interactions. We model the time course of this tetrahedron assembly process in terms of the concentration of assemblies of different sizes by a system of differential equations (shown in equations (2)) based on the above reaction equations:

$$\begin{aligned}
\frac{d[A_1]}{dt} &= -3k_a[A_1]^2 + 2k_d[A_2] - 4k_a[A_1][A_2] - 3k_a[A_1][A_3] \\
\frac{d[A_2]}{dt} &= -k_d[A_2] + \frac{3}{2}k_a[A_1]^2 - 4k_a[A_1][A_2] - \frac{8}{3}k_a[A_2]^2 \\
\frac{d[A_3]}{dt} &= 4k_a[A_1][A_2] - 3k_a[A_1][A_3] \\
\frac{d[A_4]}{dt} &= 3k_a[A_1][A_3] + \frac{4}{3}k_a[A_2]^2.
\end{aligned} \tag{2}$$

The comparison of time courses of concentrations of 4 different sizes of assemblies from 1000 simulation runs and the ODE model is shown in Figure 10-13. The parameters used in the simulation runs are N=1000, T_a=1, T_d=0.1. In each figure, the red (smooth) curve is the result of numerically integrating the deterministic ODEs and the black

(dotted) curve is the mean for 1000 simulation runs. The purple error bars correspond to mean plus/minus one standard deviation. Although the discrete event method shows a certain degree of stochastic behavior from one run to another, a close match between ODE and average simulator behavior is observed from the comparisons.

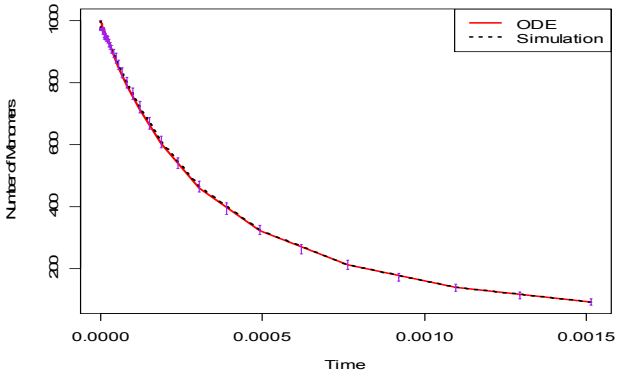

Figure 10: Time Course of Monomer Concentration

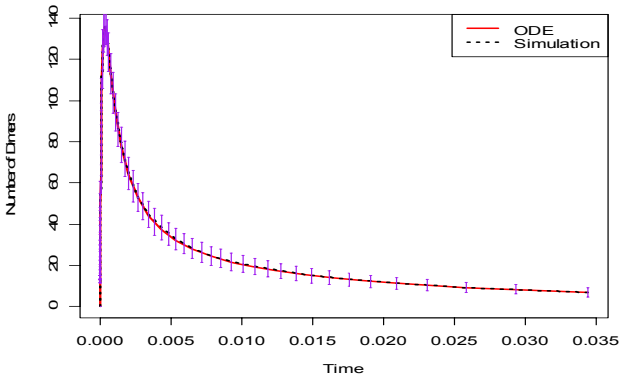

Figure 11: Time Course of Dimer Concentration

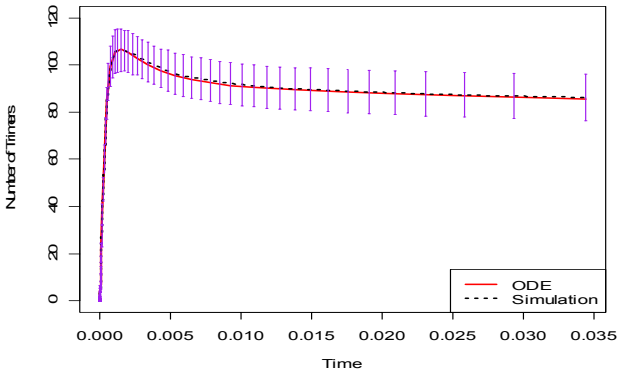

Figure 12: Time Course of Trimer Concentration

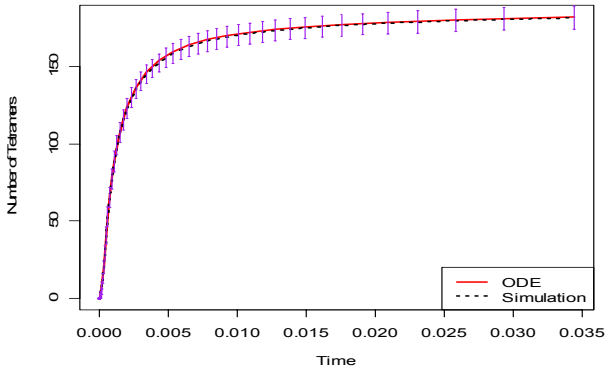

Figure 13: Time Course of Tetrahedron Concentration

4 DISCUSSION, CONCLUSIONS AND FUTURE WORK

We have implemented and tested a prototype of self-assembly simulator system with a graphical interface based on local rules abstractions and a recent time- and space-efficient discrete event simulation method. Comparisons of assembly processes of filaments and tetrahedrons between our simulation runs and corresponding ODE models confirm the simulator correctness and give us experience with the practical application of our simulator. The theoretical innovation of the simulation method combines with the flexibility and extensibility of the simulator framework to create a more complex and realistic physical model than was possible with prior work. This simulator should therefore be a valuable practical research tool for investigating the kinetics of biological self-assembly on the time and space scales typical of self-assembly reactions within living cells. The simulator is particularly well suited for large and complicated cellular self-assembly systems, such as virus capsids and the cytoskeleton.

An ongoing goal of this project is further optimization of this simulator, particular with regard to reducing initialization time and reducing memory overhead associated with graphic displays for large simulations. We are also developing a prototype XML parser to allow users to fully design and customize simulations without the need to access Java code. A principle long-term objective is to validate quantitative rates of the model using direct experimental observations, a milestone so far achieved only with simplified ODE models of capsid assembly (Zlotnick et al. 1999). While our simulator may be sufficient for this task, detailed models of specific real-world virus systems must still be constructed within the simulator.

The simulator (Version 1.2) in the form used in the present work is available at the web site `http://www-2.cs.cmu.edu/~russells/software/discrete/simulation.html`. Further updates will be released from the same site.

ACKNOWLEDGMENTS

This work was supported by U.S. National Science Foundation award #0346981. Tiequan Zhang and Rori Rohlfs were also partially supported by the Merck Computational Biology and Chemistry Program at Carnegie Mellon University. We also thank Sue Yi Chow, Blake Sweeney, Prateek Kumar and Peter Kim for their contributions to this work.

REFERENCES

Berger, B., P. W. Shor, L. Tucker-Kellog, and J. King. 1994. Local rule-based theory of virus shell assembly. *Proceedings of the National Academy of Sciences USA* 91(16): 7732-7736.

Bruinsma, R. F., W.M. Gelbart, D. Reguera, J. Rudnick, and R. Zandi. Viral self-assembly as a thermodynamic process. *Physical Review Letters* 90(24):24801-24804.

Caspar, D.L.D., and Klug, A. 1962. Physical principles in the construction of regular viruses. *Cold Spring Harbor Symposium on Quantitative Biology* 27: 1-24.

Caspar, D.L.D. 1980. Movement and self-control in protein assemblies: quasi-equivalence revisited. *Biophysical Journal* 32 (1): 103-138.

Endres, D. and A. Zlotnick. 2002. Model-based analysis of assembly kinetics for virus capsids or other spherical polymers. *Biophysical Journal* 83: 1217-1230.

Gillespie, D.T. 1976. A general method for numerically simulating the stochastic time evolution of couples chemical reactions. *Journal of Computational Physics* 22: 403-434.

Jamalyaria, F., R. Rohlfs, and R. Schwartz. 2005. Queue-based method for efficient simulation of biological self-assembly systems. *Journal of Computational Physics* 204: 100-120.

Prevelige, P.E., Thomas, D. and King, J. 1993. Nucleation and growth phases in the polymerization of coat and scaffolding subunits into icosahedral procapsid shells. *Biophysical Journal* 64: 824-835.

Reddy, V.S., H.A. Giesing, R.T. Morton, A. Kumar, C.B. Post, C.L. Brooks, 3rd, and J.E. Johnson. 1998. Energetic of quasiequivalence: computational analysis of protein-protein interactions in icosahedral viruses. *Biophysical Journal* 74: 546-558.

Schwartz, R. The local rules dynamics model for self-assembly simulation. Computer science Ph.D. thesis, Massachusetts Institute of Technology, 2000. (also published as MIT Laboratory for Computer Science Technical report MIT-LCS-TR-800)

Schwartz, R., P.W. Shor, P.E. Prevelige, and B. Berger. 1998. Local rules simulation of the kinetics of virus capsid self-assembly. *Biophysical Journal* 75(6): 2626-2636.

Schwartz, R., R. L. Garcea, and B. Berger. 2000. Local rules theory applied to polyomavirus polymorphic capsid assemblies. *Virology* 268: 461-470.

Turner, T. E., Schnell, S., Burrage, K. 2004. Stochastic approaches for modelling in vivo reactions. *Computational Biology and Chemistry* 28(3): 165-78.

Twarock, R. 2004. A tiling approach to virus capsid assembly explaining a structural puzzle in virology. *Journal of Theoretical Biology* 226(4): 477-482.

Whitesides, G. M. 1995. Self-assembling materials. *Scientific American*. September: 146-149.

Whitesides, G. M. 2002. Self-assembly at all scales. *Science* 295: 2418-2421.

Zlotnick, A. 1994. To build a virus capsid: an equilibrium model of the self assembly of polyhedral protein complexes. *Journal of Molecular Biology* 241(1): 59-67.

Zlotnick, A., J.M. Johnson, P.W. Wingfield, S.J. Stahl, and D. Endres. 1999. A theoretical model successfully identifies features of hepatitis B virus capsid assembly. *Biochemistry* 38: 14644-14652.

AUTHOR BIOGRAPHIES

TIEQUAN ZHANG is a Ph.D. student in the Department of Biological Sciences at Carnegie Mellon University. He received his M.S. in Computer Science from New Jersey Institute of Technology in 2003, and M.D. in medicine from Beijing Medical University in 1996. His Research interests are in biological system modeling. His e-mail address is tiequanz@andrew.cmu.edu.

RORI ROHLFS will attend the University of Washington Genome Sciences Ph.D. program in the fall of 2005. She graduated from Carnegie Mellon University in 2004 with a BS in computer science and a BS in biology. She can be contacted at rrohlfs@gmail.com.

RUSSELL SCHWARTZ is an assistant professor in the Department of Biological Sciences at Carnegie Mellon University. He received his B.S., M.Eng., and Ph.D. from the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. His research is concerned with the modeling and simulation of biological systems. He is a member of INFORMS, the ACM, the Biophysical Society, and the International Society for Computational Biology. His e-mail address is russells@andrew.cmu.edu and his Web address is http://www-2.cs.cmu.edu/~russells.