Modelling and Simulation of Large Assemblies of Proteins

Patrick Amar[1,2], Gilles Bernot[2], Victor Norris[3]

[1]Laboratoire de Recherches en Informatique, Université Paris Sud & CNRS UMR 8623, 15 avenue George Clémenceau, F-91405 Orsay Cedex
[2]La.M.I. Université d'Évry Val d'Essonne & CNRS UMR 8042, Tour Évry 2, 523 Place des terrasses de l'agora, F-91000 Évry
[3]Laboratoire des Processus Intégratifs Cellulaires, CNRS UMR 6037, Faculté des Sciences et Techniques, Université de Rouen, F-76821, Mont-Saint-Aignan Cedex

Author for correspondence: Patrick Amar, email: pa@lri.fr

1. Introduction

To study the assembly, the movements and the dissociation of large numbers of molecules in a virtual cell, a simulation program has been developed. The simulator is driven using a description of the model system written in a language we have also developed that does not limit the simulation program to a particular model. We present here the simulation of two completely different models: the growth of actin-like filaments in a prokaryotic cell, and the association and dissociation of proteins into large assemblies.

The program simulates a virtual cell as a three dimensional space bounded by a spherical membrane. This cell is initially filled with a population of molecules of various types. When the simulation begins, these molecules diffuse and interact according to the reaction rules described in the model. Periodically, the simulator shows the content of the cell with a 3D *OpenGL* user interface. During the simulation, the user can rotate the cell and focus on and zoom a particular area. A histogram showing the distribution of the lengths of the assemblies is also displayed. The program can also display curves showing the number of molecules of each species during the simulation.

2. Simulator description

The simulator is a stochastic automaton driven by reaction rules between molecules. Each molecule is represented by a record that includes its type, its position, a list of links to certain other molecules and other internal data. The simulator keeps track of each assembly in real time from the computer point of view.

A step of simulation (called a *generation*) is done by applying the following process:

- Choose the source molecule *S* (randomly, in order to avoid artefacts).
- Check if close enough to *S*, in a location randomly chosen *L*, there is another molecule *T*, the target.
- If so, and if a reaction rule is given between a molecule of the type of *S* and a molecule of the type of *T*, this rule is applied, according to a probability representing the reaction kinetics.
- Else, molecule *S* may move to the empty location *L*, according to a probability representing the diffusion speed.

When all the molecules involved in the cell have been processed, the current generation is completed and a new one can begin. The generation simulated time slice is set to 100 micro-seconds, which corresponds to the average time for a protein to move a distance of 10 nanometres (approximately its diameter).

To keep memory usage low, the simulated space is managed with a technique called a *hash table*. With this technique the computer memory used is proportional to the number of molecules in the simulation (and not to the size of the simulated space). The extra computer time used by the hash table is less than twice the time used to access a standard array. The proportion of this access time to the total time being less than one quarter, this extra cost is acceptable compared with the possibility of having a potentially infinite simulated space.

Another way to reduce the computer time used in this programme is to maintain two lists of molecules. The first one is the list of the *active* molecules, which can be *sources* of interaction. The second one is the list of *inactive* molecules, which can only be target of interaction. The simulator processes only the list of active molecules which can be much smaller than the total number of molecules. During the processing of a couple of molecules, the programme automatically changes the status of these molecules. In the actin example shown hereafter, the main constituent of an actin strand is the F-actin molecule, which is inactive (unless near the ends of the filament).

2.1. The rules

The simulator implements four kinds of interaction rules between two molecules: the source *S* and the target *T*:

- **Reaction**: *S* reacts with *T* to produce two other types of molecules *S'* and *T'*.
- **Association**: *S* binds to *T* to produce the complex *S-T*.
- **Dissociation**: the complex *S-T* can break and leave individual molecules *S* and *T*.
- **Catalysis**: the complex *S-T* can be transformed to *S'-T'*.

As in the reaction and catalysis rules, the association and dissociation rules can change the type of the molecules. Each rule is given a probability of execution that, on the long run, corresponds to a reaction kinetics. For the association rule, a maximum number of links can be specified.

2.2. Configuration

The simulator uses a configuration file to describe the model the user wants to simulate. This file contains four sections. The first section describes the molecules involved in the model. The second specifies the diffusion rate of each molecule. The third section describes all the reaction rules that will be applied during the simulation. The last section describes the initial population and location for each kind of molecules. Here is an example of the types of molecules involved in the simulation of the growth of actin filaments:

```
molecule
    P (200, 0, 0),                  // filamentous actin 'plus' end.
    M (0, 200, 0),                  // filamentous actin 'minus' end.
    AF (200, 200, 0, inactive),     // inactive F-actin.
    AG (200, 100, 0);               // phosphorylated globular actin.
```

With these definitions, the *plus* end (**P**) will be displayed in red, the *minus* end (**M**) in green, the filament itself (**AF**) in yellow and the free globular actin (**AG**) in orange.

In this example only the free G-actin molecules can diffuse. The filaments themselves are frozen:

```
speed (AF) = 0.0;        // diffusion speed is zero
speed (P) = 0.0;         // for the filaments.
speed (M) = 0.0;
speed (AG) = 1.0;        // high diffusion speed.
```

The following rules show the formation of the polarised dimers from two free phosphorylated free G-actin molecules:

```
AG  +  AG  ->  M(1) * P(1)    [0.05];
M   *  P   ->  AG   + AG      [0.5];
```

The right part of the first rule states that a minus end **M** can be bound to only one plus end **P**, and conversely, a plus end **P** can be bound to only one minus end **M**. The second rule is the reverse reaction, the depolymerisation of the complex, giving two free G-actin molecules.
The next rules show the growth of a filament from the plus end:

```
AG  +  P  ->  P(1)  =  AF(1)   [0.8];
P   *  AF ->  AG    +  P       [0.001];
```

The first rule shows how a free G-actin molecule can be bound to the plus end of an already existing filament. The equals sign in the right part of the rule means that the link must be aligned with the filament. The second rule shows the reverse reaction, the depolymerisation from the plus end giving one free G-actin molecule.

The next two rules show the growth of a filament from the minus end. One can notice that the polymerisation kinetics is lower than for the plus end which may lead to a linear movement of the filament towards the plus end equivalent to tread-milling.

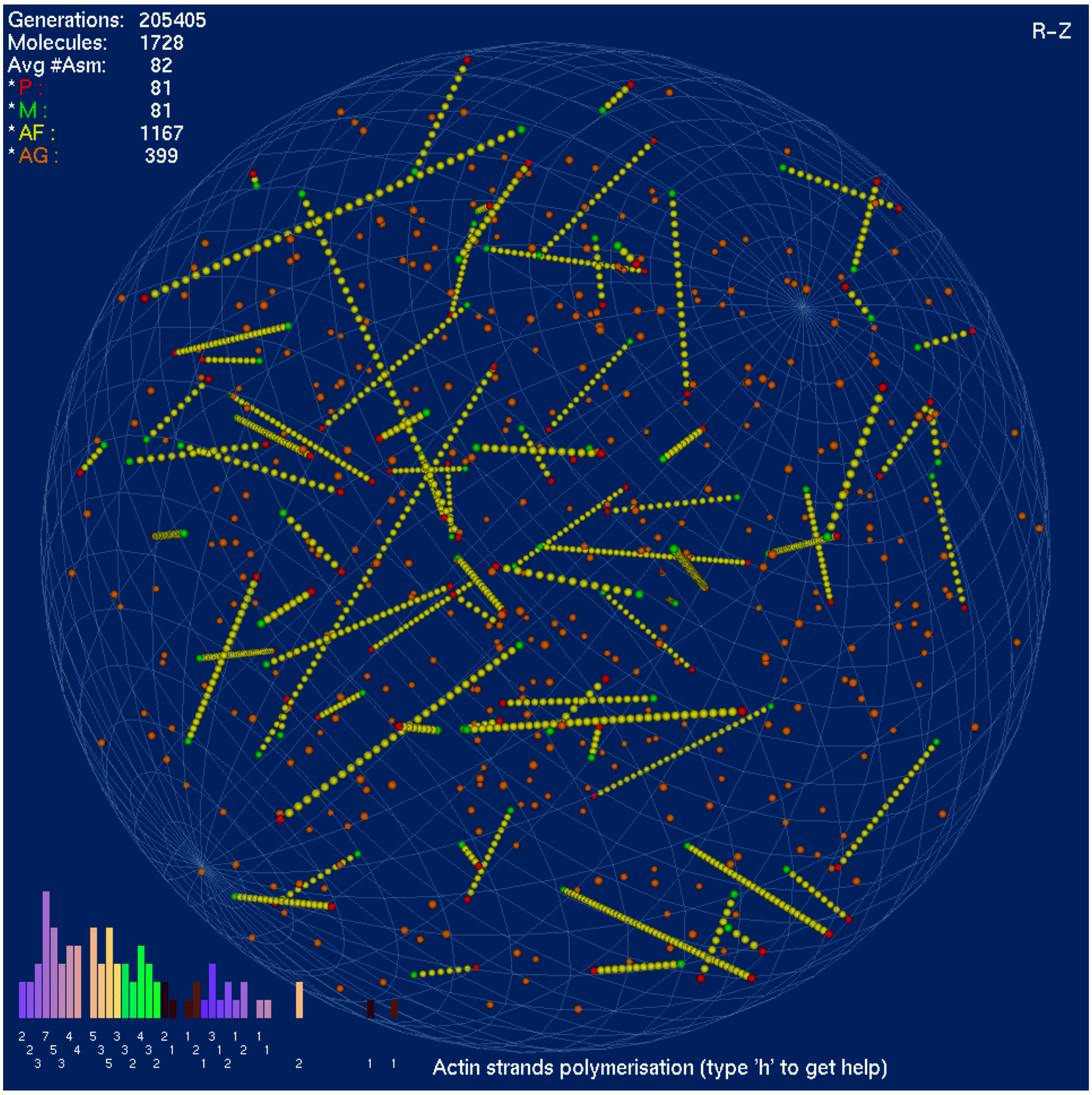
```
AG  +  M   ->  M(1)  =  AF(1)  [0.3];
M   *  AF  ->  AG    +  M      [0.005];
```

2.3. Initialisation

The simulation is initialised by the statement:

```
cube (0, 0, 0, 12, AG);
```

A cube of length *12* located at the centre of the cell *(0, 0, 0)* is filled with $12^3 = 1728$ molecules of free globular actin. This compact cube leads to a very high local concentration of G-actin. To avoid this generating artefacts, the simulator is first started in *diffusion only* mode for a few thousand generations. Then it is switched back to *reaction* mode so the molecules diffuse and react according to the rules. First, polarised dimers assemble and then the filaments grow until one end touches the membrane or an equilibrium state is reached (see Fig. 1).



```
Generations:  205405
Molecules:    1728
Avg #Asm:       82
*P :            81
*M :            81
*AF :         1167
*AG :          399
```

Figure 1: A view of the virtual cell filled with dynamic actin filaments. On the bottom left corner of the screenshot a histogram of the lengths of the filaments is displayed (see plate 2).

3. *Simulation of hyperstructures*

In addition to the rules mentioned above, the simulation language has some specific features that allow the user to study a large number of different model systems. Hyperstructures are large assemblies of molecules such as enzymes within cells. In this section we show how to model a hyperstructure in the form of a metabolic pathway in which the product of one enzyme is the substrate of the next one in the pathway. The simulation shows how hyperstructures can, on demand, assemble, work and disassemble when the simulation only specifies an increase of affinity between two enzymes in the presence of their substrate. Finally, we show how to confine an object to the cell membrane so allowing the creation of membrane receptors which can only diffuse in two dimensions in the membrane (see Fig. 2).

As in the previous example, the initial state of the simulation is obtained by switching the simulator to a *diffusion only* mode and then running it for a few thousand generations.

This disperses the cytoplasmic enzymes throughout the cytoplasm. As the diffusion speed of the membrane receptors (Enzyme 1) is very slow, they stay in roughly the same place in the membrane during this first phase of simulation. The main simulation phase begins when the simulator is switched to the *reaction* mode.

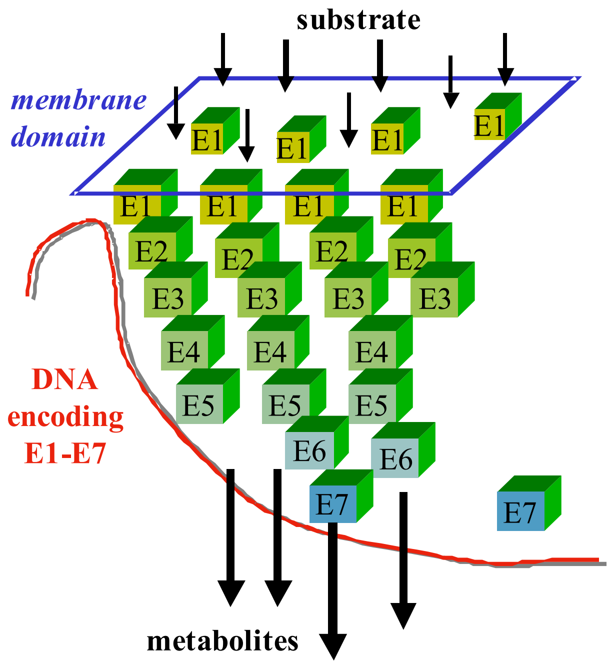


Figure 2: Formation of a non-equilibrium hyperstructure due to changes in the affinity of its constituent enzymes for one another. Enzymes E1 can only diffuse in the plane of the membrane whilst the other enzymes, E2 to E7 diffuse in the cytoplasm. The binding of a substrate, such as a sugar, to the E1 enzymes leads to an increase their affinity for one another and their assembly into an E1 domain. On binding its substrate, each enzyme in the pathway acquires an increased affinity for the following enzyme. This results in the assembly of metabolons E1 to E7 and the assembly of the hyperstructure (here, a group of metabolons). Note that transcription of the genes encoding E1 to E7 and the simultaneous translation of the mRNA may help the assembly of the hyperstructure.

With the current implementation of the programme it is not yet possible to have molecules outside the cell membrane. To simulate the fact that the initial substrate is translocated by the membrane receptor, we put this substrate directly into the cytoplasm. To keep this example as simple as possible we only use one kind of membrane receptor and a pathway of four enzymes.

3.1. Configuration

In the first section of the configuration file we declare all the molecules involved in this simulation: the membrane receptor $\mathbf{E_1}$, the enzymes $\mathbf{E_2}$ to $\mathbf{E_4}$ and the substrates $\mathbf{S_1}$ to $\mathbf{S_5}$. The *membrane* keyword declares molecules confined to the cell membrane:

```
membrane   E1 (0, 0, 250);
```

The other enzymes and the substrates are declared using the *molecule* keyword as in the previous example. The reaction rules are divided into four almost identical groups, one for each enzyme in the pathway.

The group of rules for enzyme E_n indicates that:

- enzyme E_n can fix its substrate S_n
- when E_n has bound its substrate, E_n has its affinity for enzyme E_{n+1} increased so it can be bound by it
- conversely, when enzyme E_n does not have its substrate S_n bound to it, E_n loses its affinity for E_{n+1}
- enzyme E_n transforms its substrate S_n to its product S_{n+1} , which is the substrate for enzyme E_{n+1}
- when enzymes E_n and E_{n+1} are bound together, the product S_{n+1} is transferred to E_{n+1} (and this new product is then freed from E_n and linked to E_{n+1})

Here are the rules for the beginning of the pathway, the membrane receptor E_1:

```
S1 + E1          -> S1(1) * E1(1)      [0.6];
```

This rule means that when the substrate S_1 is close enough to the membrane receptor E_1, it is captured with probability 0.6. The number between the parentheses in the right part of the rule means that the enzyme can bind only one copy of its substrate.

```
{S1}E1 + E2      -> E1 (1) * E2 (1)    [0.9];
```

This second rule shows how the *environment* of a molecule can be important in the application of a rule. The left part of the rule means that if enzyme E_2 is close enough to a membrane receptor E_1 that is already bound to its substrate S_1 (and only if E_1 is bound), then enzyme E_2 will bind to E_1 to form a complex. This is done with a high probability to model the high affinity for the two enzymes in presence of the substrate.

```
{~S1}E1 * E2     -> E1 + E2            [0.001];
```

Finally the third rule shows how enzyme E_1 loses its affinity for enzyme E_2 when E_1 is not bound to its substrate. The left part of the rule means that if E_1 is bound to E_2 but E_1 is not bound to its substrate S_1, the link between the two enzymes is broken. One can notice that the probability is very low; this is because the two molecules being linked are very close and the program gives a greater chance for this event to occur.

The two last rules of the group are used for transform S_1 to S_2 and to transfer it from enzyme E_1 to enzyme E_2.

```
{E1}S1 + {E1}E2 -> S2(1) * E2(1)       [1.0];
```

here S_1 which is bound to E_1 also binds to E_2 and at the same time is transformed to the product S_2. Each time the initial conditions are set (the left part of the rule) the rule will be statistically applied because the probability is set to one.

```
{E2}E1 * {E2}S2 -> E1 + S2             [1.0];
```

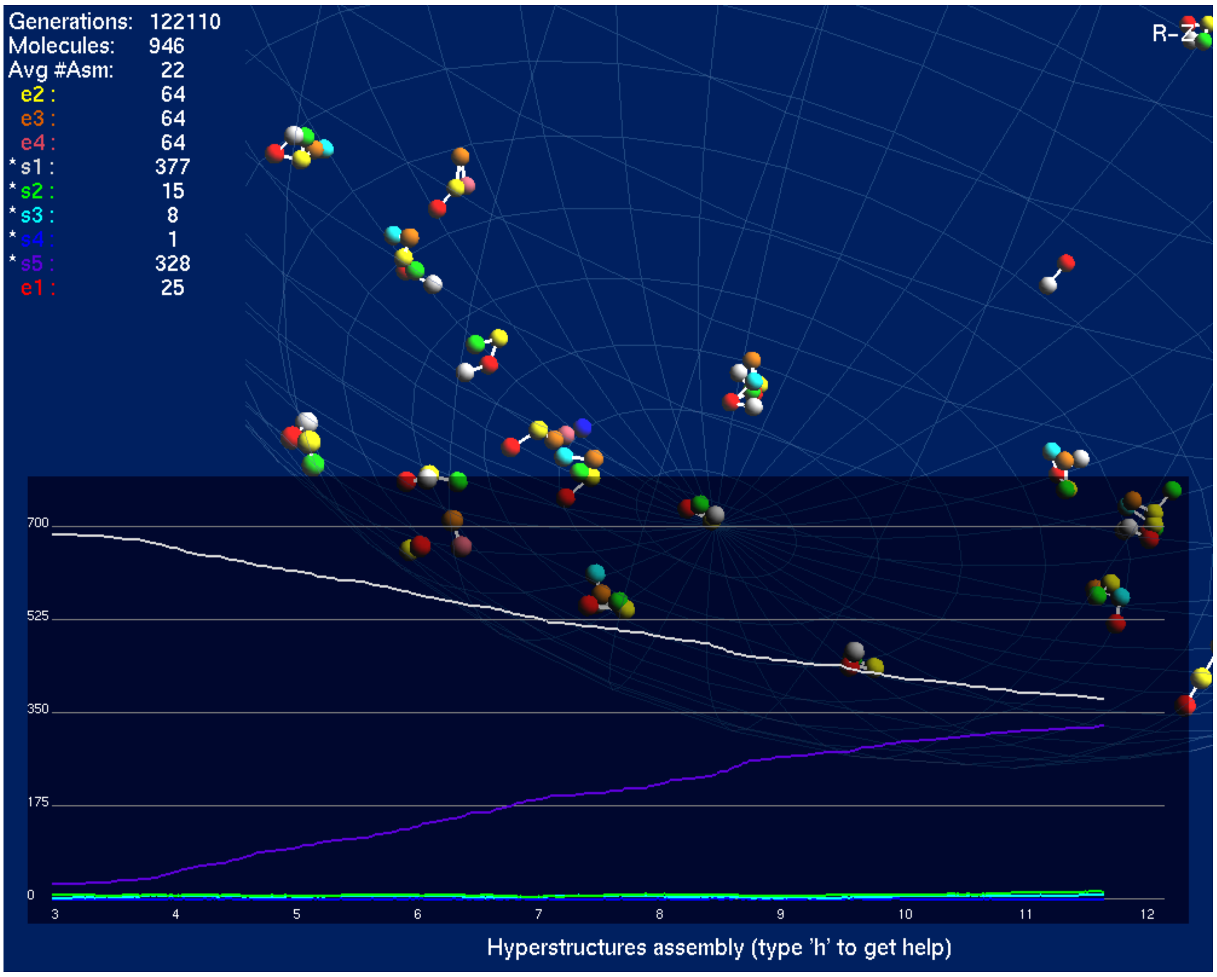
this rule is used to break the link between the product S_2 and the enzyme E_1 to complete the transfer.

With this set of five rules repeated three times (for each of the different enzymes E_2, E_3 and E_4) in the configuration file, plus a last rule to release the final product S_5 we have finished with the rule section. The initialisation section fills the cell with 36 copies of the membrane receptor, 64 copies of enzymes E_2, E_3 and E_4, and 729 copies of substrate S_1.

```
surface (E1);
cube (0, 6, 8, 4, E2);
cube (6, 0, 8, 4, E3);
cube (0, 0, 0, 4, E4);
cube (0, 0, -6, 9, S1);
```

3.2. *Simulation results*

After the first phase of diffusion to get a homogenous distribution of all the molecules in the cytoplasm, the membrane receptors bind their substrate. Then after a short period of time, we can see the first assemblies appear and quickly transform the intermediate substrates to the final product like an assembly line in a factory (see Fig. 3).



```
Generations:  122110
Molecules:    946
Avg #Asm:     22
 e2 :         64
 e3 :         64
 e4 :         64
*s1 :         377
*s2 :         15
*s3 :         8
*s4 :         1
*s5 :         328
 e1 :         25
```

Figure 3: The virtual cell with the hyperstructures linked to the membrane receptors (in red). The curves in the bottom part show the decrease in the concentration of substrate S_1 along with the increase in concentration of product S_5. The horizontal axis is graduated in seconds of simulated time (the real time is approximately 3 times slower on a standard PC). The vertical axis shows the number of copies of each kind of molecules (see plate 3).

Before all the copies of substrate S_1 have been transformed to the final product S_5, the assemblies begin to break up and finally disappear. Since these assemblies are attached to the membrane even if the total concentration of enzymes is low, the local concentration is high enough to produce S_5 at high rate.

4. Conclusion

With these two examples one can see that this simulation programme is very versatile. The efficiency of the implementation in terms of computer time is high enough to include in a future release some real time controls. These controls may include the ability to modify the reaction kinetics (the probability part of the rules) or the number of copies of each kind of molecules, etc.

Another improvement, which is in progress, is to replace the spherical membrane of the cell with a simulated membrane made with lipid molecules. The number of molecules used to make the membrane is very high, but they can be *inactive*, and so they do not use computer time. The next step is to allow deformations of this membrane, because of the pressure of the actin filaments for example.

References

Amar, P., P. Ballet, G. Barlovatz-Meimon, A. Benecke, G. Bernot, Y. Bouligand, P. Bourgine, F. Delaplace, J.-M. Delosme, M. Demarty, I. Fishov, J. Fourmentin-Guilbert, J. Fralick, J.-L. Giavitto, B. Gleyse, C. Godin, R. Incitti, F. Képès, C. Lange, L. Le Sceller, C. Loutellier, O. Michel, F. Molina, C. Monnier, R. Natowicz, V. Norris, N. Orange, H. Pollard, D. Raine, C. Ripoll, J. Rouviere-Yaniv, M. Saier jnr., P. Soler, P. Tambourin, M. Thellier, P. Tracqui, D. Ussery, J.-P. Vannier, J.-C. Vincent, P. Wiggins and A. Zemirline.
Hyperstructures, genome analysis and I-cell. Acta Biotheoretica **50**: 357-373, 2002

Borisy, G.G. and T.M. Svitkina.
Actin machinery:push-pull the envelope, Curr. Opin. Cell Biol. **12**:104-112, 2000.

Geigant E., Ladizhansky K. and A. Mogilner.
An integrodifferential model for orientational distribution of F-actin in cells. SIAM J. Appl. Math. **59**:787-809, 1998.

Kier LB, Cheng CK, Seybold PG.
Cellular automata models of chemical systems. SAR QSAR Environ Res. 2000;11(2):79-102.

Kier LB, Cheng CK, Testa B, Carrupt PA.
A cellular automata model of enzyme kinetics. J Mol Graph. 1996 Aug;14(4):227-31, 226.

Le Sceller L., C. Ripoll, M. Demarty, A. Cabin-Flaman, T. Nyström, M. Saier Jnr. and V. Norris (2000).
Modelling bacterial hyperstructures with cellular automata. Interjournal Paper 366: http://www.interjournal.org.

Wurthner JU, Mukhopadhyay AK, Peimann CJ.
A cellular automaton model of cellular signal transduction.Comput Biol Med. 2000 Jan;30(1):1-21.