computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

A GRAPH-BASED MODEL FOR THE INFECTION PHENOMENON

Cătalin Bulancea[1)] , Mitică Craus[2)]

[1)] Institute for Computer Science, Romanian Academy, catalinb@academie.is.edu.ro
[2)] "Gh. Asachi" Technical University of Iasi, Computer Engineering Department, craus@cs.tuiasi.ro

Abstract: *A graph-based model is proposed for studying interactions and evolution in infection process. There are defined and tested mutational and decisional structures for pathogen agents and a reaction mechanism for the host. MPI and C# implementations were used to make some simulations. The results have shown that artificial system evolution is closed to the evolution of the real system.*

Keywords: *artificial life, multi-agent system, message passing, graph model, leucocytes.*

1. INTRODUCTION

Artificial life applications have nature as an inspiration source. Many of the multi-agent systems have functioning mechanisms and principles inspired by natural collectivities behavior [1]. Simple life forms as seaweeds or bacteria, which have almost no individual importance, can organize themselves in complex social systems named colonies. These simple structures were the starting point for evolving to more complex multi-cellular life forms. Here is the place were we can find the basic principles of coexistence strictly connected on adaptation, interaction and selection [2], which are natural processes by excellence.

In our opinion, evolution is the most amazing natural process. Only this very complex development trend can metamorphose simple multi-entity systems, mostly of them connected by symbiosis relations, in very complicated organic systems, which have command structures (the nervous system), life support components (digestive apparatus), etc. Finally, we will name the most spectacular result of the evolution: the conscience. From simple cohesion processes between rudimentary life forms, the evolution has built the actual multi-variable mixture of electromagnetic fields and chemical reactions, which pushed us, the humans, on the climax of the living creatures pyramid.

In nature, the infection is a very complex phenomenon, strictly applicable on evolved multi-cellular life forms. We can identify the host, who is invaded with foreign, usually elementary (not necessarily unicellular) life forms. Of course, the host has some protection mechanisms, composed on white cells, antibodies, T-cells, etc. but, in some cases, these can became relatively quickly obsolete by the invader's mutation capacity. For example, the HIV virus should be a very simple problem for the human immune system, if he wouldn't have the actual extraordinary mutation capacity, which makes him very hard to be identified by immune system detectors.

From a kinetic point of view, most infections presume massive multiplications, and propagation phases. There is a computational fight between organism's making new antibodies capacity and invader's mutation possibilities. As we can see, both sides are searching for the optimum in separated evolution processes.

In this paper we describe a multi-agent system with two separated populations who interact inside an environment which has the infrastructure modeled as a graph. The graph's nodes can be seen as points with maximum concentration of resource (we name resource substances indispensable for life process, which are consumed by pathogen agents, usually resulting toxins). The nodes are interconnected using communications channels, which can be, in real organisms, capillary blood vessels or, much simpler, communications paths. The entities (antigens and antibodies) are modeled as messages, which can travel between nodes.

Initially, the infection is located in few nodes. The graph should evolve to supra-infected stage, if the reaction is not efficient, or to clear stage, in the contrary situation. If no "living nodes" remain, the graph becomes dead.

2. PROBLEM STATEMENT

In nature, infectious agents are primitive life-forms (usual viruses and bacteria). They have considerable reproduction capabilities in a favorable environment. They are also exposed to mutations. The result is a better competition between them and the immune system.

In few cases, the infection is focused on a single point. Usually, the pathogen agents are grouped around maximum concentration resource (substances needed for their metabolism) points. If the resource is consumed, they will migrate to another places were it is possible to find resources. In all this time, the immune system is trying to reduce, and if it is possible to destroy them through anti-viral self-made agents (leucocytes, T-cells, antibodies).

3. MATHEMATICAL MODEL

In this section will be described the environment, the mutation mechanism of the viral agents and the reaction.

In order to model the environment infrastructure, we consider a graph. As we've already mentioned, the nodes can be considered the maximum concentration resource points. In these points, this substance(s) concentration gradient is zero or doesn't exist. Communication channels, represented by the edges of the graph, link such points.

Some functions are assigned to the system and they will measure stability and will correlate this with reaction efficiency..

We can define a graph-based model for the infection problem, as it follows:

A graph $G=(V,E)$ represents the environment infrastructure. V is the set of the nodes ($|V|=n$) and E is the set of the edges. ($|E|=m$). Initially, a small number of infectious agents are placed in a single (or few) node(s).

Each node can host a certain number of entities, different from one node to another. We call this the *node capacity*. So, a node j has an associated capacity c_j.

The infection entities have associated a set of characteristics: *acidity (a), life (l), sensibility (s), reproduction_value (rv)*. These characteristics can be transferred to the next generation using chromosomes *(ch)*, composed by genes. The genes can take numerical values and each of them represents an entity characteristic.

In reality, genes are exposed to mutations, which can change them and randomly modify characteristics. It can be said that mutations are the main engine of evolution. We have modeled this process using a mutation operator, which is defined as it follows:

$$m : I_{chromosome} \rightarrow D_{genes} \quad (1)$$

where m is the mutation operator, $I_{chromosome}$ is the domain of the chromosome indexes and D_{genes} is the domain of the gene values. This operator can be applied either in the reproduction process or in some special situations given by the local environment toxicity or hostility. Let us denote by ch_i the i-th chromosome belonging to the entity e. For the chromosome ch_I, the operator m can be applied to one or more genes. m is not a common multi-variable function. It will randomly select genes (one or more) over which will be applied the mutation [3]. There is a restriction in considering $rv<s$ ($rv<<s$ is better). This can be explained by the fact that the biological entities (even they are or not pathogen agents) rarely reproduce themselves in a hostile environment (with no food). Reproduction is modeled using a *multiply* operator, which is unary, acts over a single entity (parent entity) and produce a child. In some situations multiplication is accompanied by mutations. This means that children entities are not accurate copies of their parents and they could have some different characteristics, which can make them stronger (or weaker) than their parents. Consider a bacteria population who is sensible to a certain antibiotic. There can appear an individual, which is immune to this substance. He will survive and he'll find a more favorable environment for reproduction

The decision structure of the pathogen agents is very simple in nature. Simple life forms as bacteria just follow a growing concentration gradient of the resource, while viruses are moving randomly most of the time (they reproduce themselves using host cells).

The decision function is using a Monte-Carlo strategy. This function is generating a partial random number d (partial because there are some dependencies on local environment, for example current resource quantity). We can consider three stages:

1. If the value d is lower than first threshold which is considered the reproduction value, the entity reproduces itself (multiply operator is applied);
2. If the value d is greater than first threshold and lower than secondary threshold (sensibility value) the entity is staying in current node and waits for new events;
3. If d is greater than secondary threshold that means this place is not friendly, or has no food (resource) and entity will migrate to another node.

The migrations mechanism can be defined in two ways as it can be seen in Fig.1 and Fig.2.

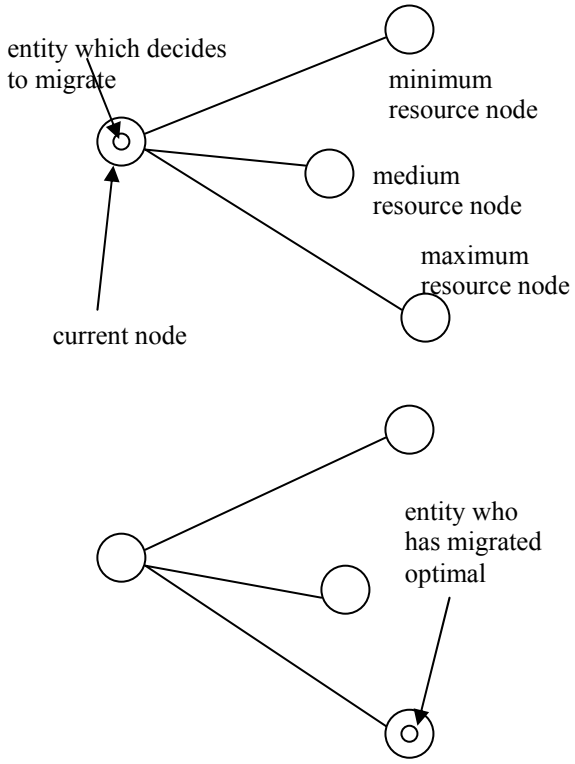

Fig. 1 - Deterministic migration process

In Fig.1 the entity (pathogen agent) is migrating following an optimal way. It chooses the node with maximum concetration of resource. This capability is specific to the more evolved unicellular organisms with self movement capacities (scourges).

The secondary way of migration is specific to rudimentary life forms like viruses or primitive bacteria forms (Fig.2). They do not feel any concentration gradient and are moving themselves in a random way. For this kind of migration we developed a strategy to minimize the probability that the migrating entity to turn back, in the origin node.

Until now, we've discussed only the infection component of this system. The other part, the reaction, is designed to simulate parts of the immune system reactions in a normal infection case.

The natural immune system has an impressive number of components. Many of them haven't been discovered yet. One of the most amazing components is represented by the B-cells or memory cells which can store information about previous infection agents, so the efficiency of the immune system is increased many times. The main limitation is resulting from the only 10^8 combinations that can be stored in memory cells, as compared to 10^{16} generating possibilities.

Let's take the white cells (leucocytes). They have some external receptors (tolls), which become activated on pathogen agents. When these receptors touch a certain cell, they are identifying a pattern in constituent substances disposition on external cell membrane.

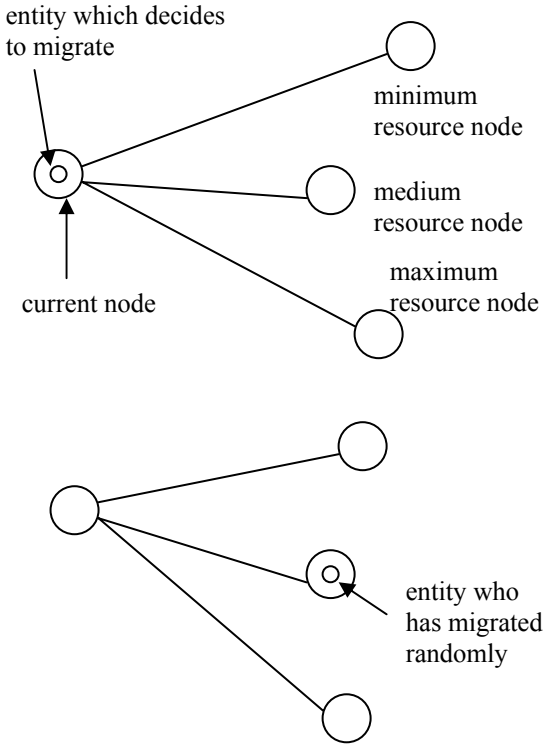

Fig. 2 - Random migration process

If this cell is recognized as foreign, it is destroyed otherwise nothing is happening. These anti-infection agents are generated in such manner to not match with body cells, named *self*. If this is happening, then the white cell (who is still in a preliminary stage) is destroyed. This generation mechanism is shown in Fig.3.

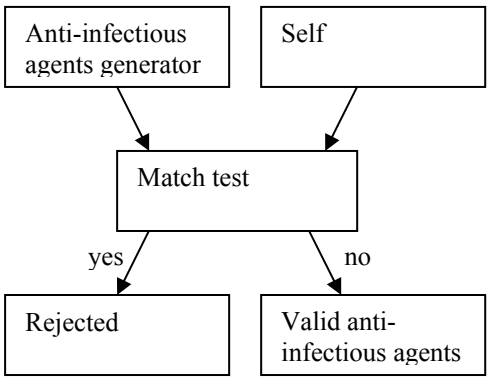

Fig. 3 - Anti-infectious agent generation process

Each anti-infectious agent has a number of receptors (tolls); these do not have to match with another system elements (in our case – other white cells entities). Those agents are generated in some nodes, named *active nodes*, only in the case of a local infection; they were not provided yet with a

migration capacity; their life-time is limited and they are generated with resource cost.

Only this part of reaction has been simulated in our graph-based model.

4. EXPERIMENTS

In order to test our model we have used the message-passing paradigm. The nodes were considered processes and the entities were defined as messages exchanged by the nodes.

We have built a C# simulator, with a statistic module and we have made some tests.

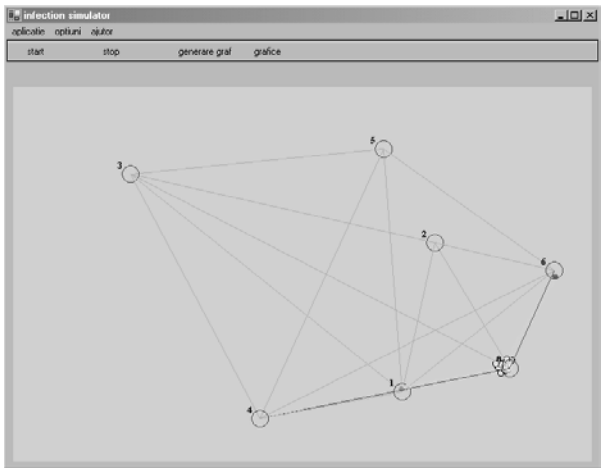

Fig. 4 - Main window of infection simulator

The generic algorithm for each node, implemented as a separated process is described as it follows:

```
for (each node) do in parallel
        if (entity.arrived()) then
                population.increase()
        endif
        if (node.state=infected) then
                node.activatereaction()
        endif
        //lifecycle
        for (each entity in node) do
            if (entity.life=0) then
                entity.dies()
                population.decrease()
            endif
            entity.decreaselife()
            if (entity.type=white cell) then
                entity.interract()
            else
                entity.decide()
                if (decision=multiply) then
                    entity.multiply()
                    population.increase()
                else if (decision=stay) then
                    entity.stay()
                else
                    entity.migrate(neighbors)
                    population.decrease()
                endif
            endif
        endfor
endfor
```

In Fig.4 is presented the main window of our C# simulator. Here it can be seen "live" the infection evolution and the reaction response.

The infectious agents either are traveling from one node to another or are staying in current node and multiply themselves.

The results can be seen using a special „statistic" module after the simulation is over (Fig.5). This module allows making comparisons between antigen population and reaction intensity. The energy variations can also be visualised.

This simulator was designed to be well balanced. Different initial data sets can guide to different final results (supra-infection or total elimination of infectious agents).

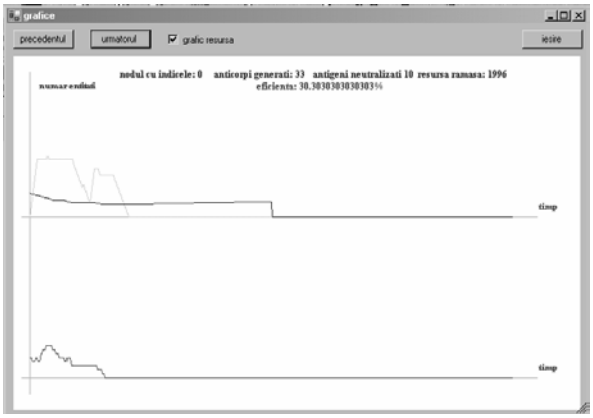

Fig. 5 - Graphics from the statistic module

There were made ten simulations for every mutation rate (the mutation probability to appear after more reproductions). The correlation between mutation rate, simulation time and graph state at the end of simulation are summarized in Table 1. The numbers from the table are average values of the ten simulations results.

Table 1. Simulation results

Mutation rate	Simulation time	Graph damage
0.1	450 cycles	12%
0.3	520 cycles	15%
0.5	670 cycles	28%
0.7	650 cycles	34%
0.9	800 cycles	42%

It can be observed that a great value of the mutation rate makes the infectious agents stronger against the reaction. Over a mutation rate value of 0.7, the supra-infection cases (total graph destruction) have a greater density.

This phenomenon is present in natural world too. There was observed that highly mutating viruses or bacteria infections are harder to eliminate by the natural immune system.

5. FUTURE WORK

The improvement of the reaction model is a future work. There are many unexplored aspects concerning the host reaction.

Other future work is to make our graph-based model more closed to some real type of infection and to test it on some real data sets.

6. REFERENCES

[1] Y. Shi. and R. C. Eberhart, Parameter selection in particle swarm optimization, Evolutionary Programming VII: Proc. EP 98, Springer-Verlag, New York, 1998, pp. 591-600

[2] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975

[3] D. Dumitrescu, Genetic Algorithms and Evolutive strategies, Microinformatica, Cluj-Napoca, 2002, p.53

Catalin Bulancea *was born on 26 august 1978.*

Education: 2002 -2003, Master student in "Distributed Systems" Department of Automatic Control and Computer Engineering, Technical University "Gh. Asachi", Iasi; 1997 - 2002 , Faculty of Automatic Control and Computer Engineering, Technical University "Gh. Asachi", Iasi.

Computers specialization – software.

Employment: 2002 - present, research fellow at Romanian Academy, Computer Science Institute, Neural Nets and Image Processing Department.

Computer skills: Artificial Intelligence - OCR Implemented in JAVA (using neural nets); ALife – simulations.

Other - Load Balancing, Leader Election and Graph Spanning Tree algorithms implemented in MPI, Programming knowledge - C, C++, Visual C++, JAVA, MPI, SQL/PL SQL – beginner.

Foreign Languages: English – good, French – medium.

Areas of Interests: Artificial Life simulations including genetic and behavioral mechanisms.

Mitică Craus.

Education: 1975 – 1979 : Student at the Faculty of Mathematics - Computer Science Department of the "Al. I. Cuza" University of Iasi.

1999 : Ph.D. in computer science.

Present job: Associated professor at the Faculty of Automatic Control and Computer Engineering, Department of Computer Engineering.

Computer skills: Data structures, algorithm design (sequential, parallel and distributed), C/C++, Java, PASCAL and FORTRAN programming.