MAiS: Exploiting JADE as a Multi-Agent simulator of the Immune System

Sanchayan Bhunia[1], Angelo Ferrando[1][0000-0002-8711-4670], Viviana Mascardi[1][0000-0002-2261-9926], and Chiara Vitale[2][0000-0002-2865-1281]

[1] Department of Computer Science, Bioengineering, Robotics and Systems Engineering (DIBRIS), University of Genova, Italy
bhunia.sanchayan@gmail.com, angelo.ferrando@unige.it, viviana.mascardi@unige.it

[2] Department of Experimental Medicine (DIMES), University of Genova, Italy
chiara.vitale@unige.it

Abstract. The immune system is the second most complex biological system after the brain. It consists in millions of cells, of various nature, interacting amongst them to keep the organism safe from external enemies (pathogens), such as viruses and bacteria. To better understand how the immune system works, and how it reacts to certain diseases and cures, simulations have been proposed over the years. Amongst them, we may find agent-based ones where the organism's actors, like cells, antibodies, viruses, and so on, are represented as agents. In this paper, we present the initial design and development of an agent-based simulation of the immune system using a well-known agent framework, JADE. We present the engineering choices we made and the instantiation of some steps of the secondary immune system response. We discuss the implementation in JADE, and we present some experimental results.

Keywords: Agent-based Simulation · Immune System · JADE.

1 Introduction

The immune system of any multi-cellular organism is a very complex system with a lot of different T-cells playing important role to keep a body free from infections, such as those due to viruses. The goal of the virus is to hijack a cell and use its resources to make a copy of itself (replication) and to spread those replicas into other parts of the body. The immune system helps an organism to fight back the infection by searching for any virus signature moving from one cell to another. The immune system in general consists of multiple different cells with different behaviours: some of them immediately activate and contribute to eradicate the infection by different and complementary mechanisms, such as Phagocytes, T-cells, B-cells; some of them remember the signature of an infection, such as B-cells and Memory T-cells, and are essential to prevent similar infection in the future.

The goal of this paper is to model some of the components of the immune system and as interacting agents using the Java Agent DEvelopment Framework (JADE [2]) and simulate a situation when the immune system is under attack from a virus. Agent-Based Model & Simulation (ABMS) is a well-known research area focused on simulating systems following a bottom-up approach, where agents are used to describe the actors of the simulation (for further readings on ABMS techniques a review can be found in [18]). Specifically, this work is about the engineering of an agent-based simulator in JADE. JADE is a well-known agent framework based in Java and used in many industrial applications for its being natively distributable, and having a shallow learning curve. We present an initial engineering and development of a small part of the immune system in JADE, namely the secondary immune response carried out by specialised Memory T-cells. We show the engineering process that brought us to the development of the digital twins cells as JADE agents, and how their behaviours have been coded as JADE behaviours. We recognise the part currently tackled in our integration in JADE is limited, and an over-simplification of how a realistic multi-cellular organism's immune system behaves; however, we decided to focus on the engineering aspects of finding suitable mappings amongst biological entities (cells) and digital ones (agents). Above all, we are interested in presenting the skeleton of an agent-based simulator, called MAiS (Multi-Agent immune System), which is going to serve us as a base for future developments and extensions that will be considered to properly model the actual immune system. Finally, we point out that no simulation of the immune system has ever been proposed in JADE.

The paper is structured as follows. Section 2 introduces background knowledge on JADE and immune system to make the paper as widely accessible as possible. Section 3 presents the engineering decisions made in the MAiS development, as well as the main reasons for implementing the latter on top of JADE. Section 4 presents the agents used in MAiS to represent the biological entities and how such agents work with each other. Section 5 reports the results obtained by carrying out experiments with MAiS. Section 6 positions the paper w.r.t. the state of the art. Finally, Section 7 concludes the paper and points out future directions.

2 Preliminaries

2.1 JADE

JADE[3] [2] is a widely used agent framework. By being based upon Java, it is easy to learn and to integrate to existing solutions. JADE is structured around the idea of agents, behaviours, and containers. Agents follow the standard meaning, as main actors of the system. Each agent runs on a different thread, and it communicates with the other agents through FIPA[4] compliant messages. Behaviours denote how the agents act, and how they achieve their goals. Containers

[3] `https://jade.tilab.com`
[4] `http://www.fipa.org`

represent the abstract environment where the agents live, and can be distributed over multiple machines. It is important to note, that agents can move amongst different containers. This is going to be exploited in the paper to simulate the passage of viruses and lymphocyte amongst different cells. Note that, since the containers can be deployed on different machines, the agents not only move, but their computations move as well. This aspect is of paramount importance in case the system to model becomes too big to be handled by a single machine; as it would happen for a realistic immune system comprised of millions of cells.

2.2 The Secondary Immune Response Scenario

The immune system is our second most complex system, after the brain. It involves many different actors (*i.e.*, cells of our body's tissues, pathogens like bacteria and viruses), each one with different features, capabilities and objectives. Depending on the scenario, and the current state of our organism, we may find ourselves with a complete different set of cells involved in defending us, and that carry out different tasks.

In this section we consider a simple – but correct – scenario, that will be used as the case study for the initial design and implementation of MAiS.

1. Some time ago, Alice got an infection from a virus V, which infected cells of tissue TIS (target cells). The infection fired a reaction from the immune system that, besides successfully fighting V, also activated the generation of Memory T-cells specific for V, that we name $MemT(V)$. Memory T-cells are a subset of T Lymphocytes and their primary function is augmented immune response once the pathogen that initially caused their generation (V in this scenario) attacks the body again. Memory T-cells may 'wander' in the blood and in tissues close to those that had been infected by the pathogen, even 30 years after the first infection.
2. Alice comes into contact with V again.
3. V enters her body via her mouth and tries to infect cells of tissue TIS again. It may succeed in this attempt with some infection success rate ISR. Let us suppose that one cell $C(TIS)_a$ becomes infected. Two events take place:
   (a) $C(TIS)_a$ becomes a virus-making factory: V can in fact use it to move to 'infectable neighbouring cells', and infect them. The amount and density of infectable neighbouring cells depends on TIS, on the point the virus entered the body, and on other factors.
   (b) $C(TIS)_a$ can be recognized as infected thanks to a peptide (an antigen) on its membrane surface, acting as a manifesto for its health state: when a cell is infected with a virus, it has pieces of antigens of that virus – the virus 'signature' – on its surface, not present when the cell is healthy.
4. One Memory T-cell trained to recognize V, $MemT(V)_b$, that was wandering close to $C(TIS)_a$, perceives pieces of V's antigen on the cell's membrane[5].
5. $MemT(V)_b$ kills $C(TIS)_a$: this is the only way to stop the proliferation and spread of V.

[5] Memory T-cells $MemT(V)$ recognize *only* antigens of V, since they are virus-specific.

6. Depending on TIS, cells may be re-generated (cellular replication) to cope with the fact that some amount of $C(TIS)$ infected cells were killed; replication may accelerate to re-establish some stable number of cells, or may not take place at all, depending on TIS. For example, cells of the nervous systems cannot be replicated (perennial cells), hepatic cells can, but only as a response of a serious damage and up to some extent (stable cells), blood cells are continuously replicated (labile cells). If the damage is too vast, even tissues with labile cells may not be able to restore at the healthy situation holding before the virus infection.

3 The engineering of MAiS

MAiS has been engineered to be a highly scalable and distributed agent-based immune system simulator. Even though in this paper we only focus on a simple sub-process of the immune system's response, we tackle all the fundamental aspects of its full engineering. Specifically, we address how to represent the main immune system's components inside JADE. As we mentioned in the introduction, the main actors of the immune system are the cells. A cell does not only denote the building block of any organic system, but it also represents the point of attack for viruses as well (for replication). Because of its importance, we started the engineering of MAiS from the cells.

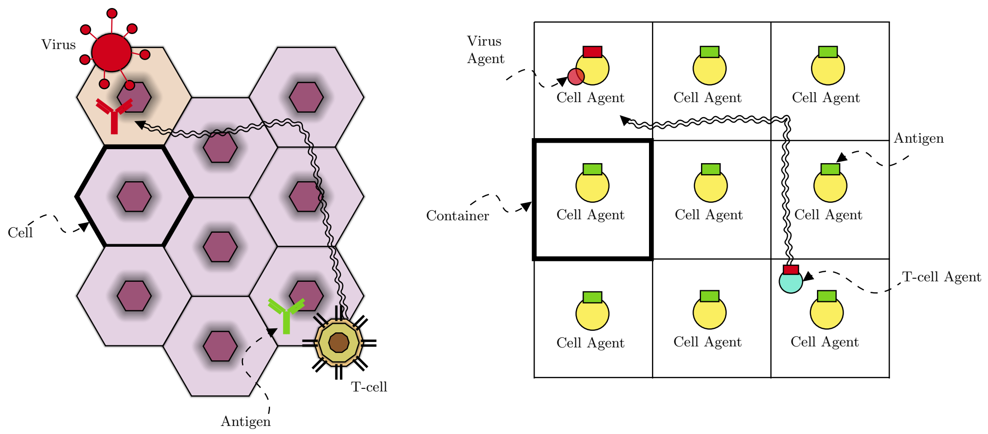

Fig. 1. Schematic of how the biological components (left) are mapped into MAiS (right).

Figure 1 graphically shows the engineering of the immune system's pillar components in MAiS. On the left, we may find a schematic representation of a region of cells in the organic system. While, on the right, we may find the counterpart in MAiS, where each cell is represented as an agent (Cell Agent).

One important aspect to clarify is the locality of cells. Indeed, each cell is not an island, but is connected to other cells. This aspect has been mapped into MAiS by adding each Cell Agent inside a JADE container. Then, to maintain the positioning of the cells, the JADE containers are stored keeping the information about their neighbour containers (*i.e.*, neighbour cells). This can be schematically represented as a grid of containers (as shown in Figure 1). We decided to specify a cell as the combination of a Cell Agent and a JADE container for two main reasons: (i) to exploit the agents' mobility feature of JADE; (ii) to exploit the intrinsic distribution of containers in JADE. Thanks to (i), if we denote a cell as a container, we can natively implement in JADE the movement of T-cells and viruses through the movement of T-Cell Agents and Virus Agents between containers. Such feature is available in JADE, and allows the agents to move between containers in a transparent way. This is very important both for T-cells, that need to move around the system searching for viruses, and for viruses as well, that need to replicate to neighbour cells. On the other hand, thanks to (ii), by denoting the cells as containers, we can distribute the latter on multiple machines. By doing that, we can exploit the computational power of cluster of machines as well as parallel computing. This aspect is of paramount importance when the immune system to model starts growing; since it would be increasingly hard for a single machine to handle millions of cells (*i.e.*, millions of agents). These two features make JADE a suitable candidate to specify distributed simulations of the immune system.

4 Agents

Now that we have clarified the topology of the immune system representation in JADE as a grid of containers, we can focus on the main actors of the immune system and their agent representation.

We have three different typology of agents: Cell Agents, T-Cell Agents, and Virus Agents.

4.1 Cell Agent

A Cell Agent, as the name suggests, represents a cell in the body of a multicellular organism. It lives inside a JADE container and has an identifying peptide (*i.e.*, the antigen) which is used by the immune system to verify the status of the cell (healthy or infected). In case a cell is detected as infected, the immune system reacts by killing such cell (by means of T-cells). The entire "identify and kill" process is necessary to avoid the propagation of the infection to other cells.

To represent this behaviour in MAiS, each Cell Agent stores its antigen information, and allows other agents in the system to interact with it. Such antigen is expressed by the cell antigen as as string storing the peptide ammino sequence. T-Cell Agents can read such antigen to recognise if belonging to a known virus, or, Virus Agents can replace cellular antigen with an own viral antigen to simulate the binding of the virus with the cell membrane (action performed by the virus for replicating itself through the cell). We discuss these two mechanics when we present the corresponding agents in the following sections. The identification carried out by a T-Cell Agent is obtained by checking the corresponding string antigen for a complete match (*i.e.*, the antigen peptide sequence has to be 100% equal to the one known by the T-cell).

In here, we present the behaviours of the Cell Agent, and how it communicates with the other agents in the MAS.

Let us focus on the behaviour first. A Cell Agent enacts three different behaviours, which correspond to three different states.

Healthy: At the beginning of the simulation, the Cell Agent is healthy and has not been infected by any virus, yet. In this state, the Cell Agent listens for communications from any possible agent, including Virus Agents. This listening phase is obtained in JADE by the use of a Cyclic Behaviour[6], where the agent accepts communications from other agents in its container (*i.e.*, agents that are local to it). In this phase, a Virus Agent can move to the Cell Agent's container, and send a message to the latter triggering the infection. This causes the Cyclic Behaviour to be removed, and the Cell Agent's antigen to be modified according to the Virus Agent needs.

Infected: After a Cell Agent has been infected by a Virus Agent, the only Cyclic Behaviour which remains listens to T-Cell Agents (since cannot be infected anymore by being already infected the behaviour listening for the Virus Agent is removed after the previous step). Such behaviour listens for messages from any T-Cell which is located inside the Cell Agent's container. When this happens, the T-Cell Agent asks the Cell Agent to communicate its own antigen, and if the antigen (which was previously modified by the virus) is recognised by the T-Cell Agent, the Cell Agent (along with its residing virus) is killed.

Regenerating: After the Cell Agent has been killed by the T-Cell Agent, the Cell Agent can start a process of rebirth (induced by neighbour cells). Hence, a new Cell Agent is created to substitute the previously killed one. After that, the cell is considered healthy again and the process can restart.

We want to linger a bit longer on the regeneration phase. In particular, we have to point out that the act of regenerating a new cell is not a trivial process, and is not always possible in real life scenarios. Indeed, not all cells in the human body can be regenerated; an example are brain and heart cells, which once they are lost, for any reason, are not regenerated by the body (causing the affected organ to function in a more limited way, *i.e.*, never fully recovering). In MAiS, we are currently focusing on cells that can be regenerated, according to some regeneration factor (which of course would depend on the typology of the cell).

The other fundamental aspect of the Cell Agent is the communication with the other agents in the system. As we mentioned before, depending on which state the Cell Agent currently is, the communication may vary. When the Cell Agent is healthy, the infection from a Virus Agent can be triggered by message

[6] A behaviour which is cyclically executed by the JADE agent (until is not removed).

exchange. The same reasoning is followed with the T-Cell Agent as well, which can trigger the destruction of the cell and the virus by message exchange. Specifically, at the Cell Agent level, we may find three different kind of messages being exchanged:

Virus→Cell: When the Virus Agent enters the Cell Agent's container, it sends a message to the Cell Agent informing the latter of the attack and requesting the Cell Agent to change. This is obtained in JADE by exchanging an `REQUEST` message[7] over a specific communication channel, named `Update_Antigen_Peptide`.

Cell↔T-Cell: When a T-Cell Agent enters the Cell Agent's container, it sends a message to the Cell Agent asking to verify the cell's antigen. This is obtained in JADE by exchanging a `REQUEST` message over a specific communication channel, named `Antigen_Peptide_Verification`. Upon such request, the Cell Agent sends a message over the same channel informing the T-Cell of the current antigen peptide sequence.

T-Cell→Cell: Finally, when a T-Cell Agent has killed a Cell Agent (along with its virus), it sends a message to the Cell Agent asking to start the regeneration process. This is obtained in JADE by exchanging an `INFORM` message over the `Start_Regeneration` communication channel (which triggers the cell regeneration).

4.2 T-Cell Agent

The T-Cell Agent is the agent which executes the task of identifying and killing infected cells. Upon initiation, the T-Cell Agent is created inside a random container. This is a mobile agent which can move from one container to another one. Such movement is allowed only between containers that are neighbours on the grid (as presented in Section 3). After moving to a new container, the T-Cell Agent asks the Cell Agent present in that container to verify its antigen peptide sequence. Upon retrieval of the antigen peptide, it performs an identification process to detect whether the cell has been infected by a virus known by the T-Cell Agent. If that is the case, it kills the Cell Agent present in that container and sends a request to the corresponding Cell Agent to start the regeneration process. After completing the killing step, the T-Cell Agent restarts moving to a new neighbour container and it repeats its Cyclic Behaviours.

As we did for the Cell Agent, let us know focus on the behaviours of the T-Cell Agent. In more detail, a T-Cell agent enacts three different behaviours, which correspond to three different states.

Searching: The T-Cell Agent's standard behaviour is to move between neighbour cells. This is obtained in JADE by instantiating a One-Shot Behaviour[8] which makes the agent move from the current container to a neighbour one. This is obtained by calling the native JADE method `doMove()`, which moves the agent into a destination container (passed as argument). The movement

[7] A message with `INFORM` FIPA performative.

[8] A behaviour which is executed by the agent only once.

is completely handled by the JADE framework, and makes the movement amongst different cell containers transparent to the T-Cell Agent.

Checking: After completing the movement and having reached a new cell to inspect, the T-Cell Agent communicates with the Cell Agent inside the currently visited container asking for its antigen peptide sequence. This is implemented as a One-Shot behaviour and consists in not only the communication step with the Cell Agent, but in the verification of the antigen peptide sequence as well. After such identification is completed, if the T-Cell detected the antigen peptide as the one of the virus, then the T-Cell Agent carries on with the killing of the Cell Agent (along with its residing virus). Otherwise, the T-Cell Agent restarts moving as explained in the previous step, since the currently analysed cell is healthy (or is infected by a virus that is not recognisable by the T-Cell Agent[9]).

Killing: In case the antigen peptide sequence of the analysed Cell Agent actually matches the one recognised by the T-Cell Agent, there is an infected cell to kill (to stop the replication). In JADE this is obtained by instantiating a One-Shot Behaviour calling the JADE native method `kill()`, which kills an agent in the MAS. After the Cell Agent, and its Virus Agent, have been killed, the T-Cell Agent communicates to the Cell Agent in the container asking to start the regeneration process. After that, it restarts moving.

The communication aspects which are relevant for the T-Cell Agent are the messages exchanged with the Cell Agent about the verification of the antigen peptide sequence. We have already explained how such message passing works in the previous section, where the T-Cell agent sends a `REQUEST` message over the `Antigen_Peptide_Verification` communication channel to the Cell Agent, and waits for the corresponding `INFORM` message containing the sequence to verify.

Figure 2 reports a Sequence Diagram on the interaction between a Cell Agent and a T-Cell Agent. Such diagram schematically presents the steps carried out by a T-Cell Agent w.r.t. a Cell Agent, as we explained previously. Note that, the diagram is very abstract. To be more precise, the `doMove()` method requires the name of the target container in advance (such information is stored inside each Cell Agent, which keeps track of both the current container, and the neighbour ones). The rest of the methods and messages used in the diagram have been presented before, and their meaning is straightforward.

4.3 Virus Agent

The Virus Agent mimics a virus in real life whose goal is to replicate itself exploiting the resources of the host cells. Differently from the T-Cell Agent, the Virus Agent is not a mobile agent. In fact, it can only move by replicating itself to the adjacent cells. Like it happens in nature, the Virus Agent has a replication factor, which defines how many copies the Virus Agent can make of

[9] We have to keep in mind that a T-Cell Agent is specific for a single Virus Agent, it knows how to recognise and destroy only that kind of virus; all other viruses in the system are invisible to it.

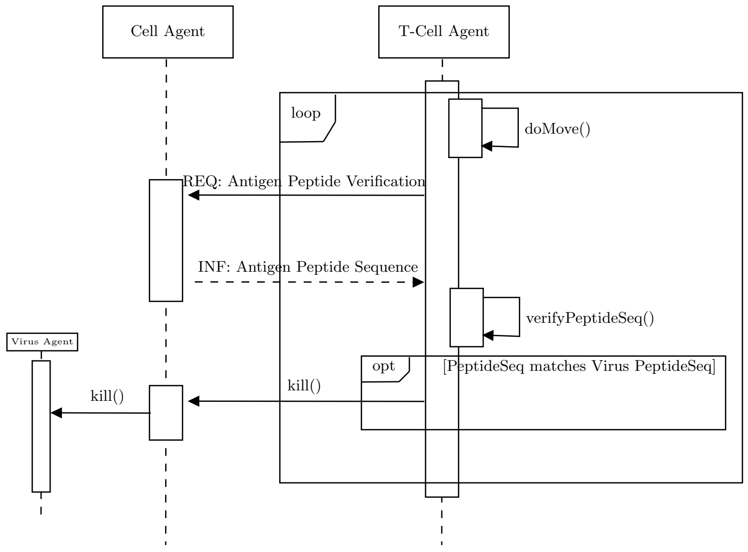
Cell Agent
T-Cell Agent
Virus Agent
loop
doMove()
REQ: Antigen Peptide Verification
INF: Antigen Peptide Sequence
verifyPeptideSeq()
opt
[PeptideSeq matches Virus PeptideSeq]
kill()
kill()

Fig. 2. Sequence diagram of the interaction amongst a Cells Agents and T-Cell Agents.

itself before destroying the host cell. Such replication factor is an input parameter of MAiS, and it can be customised to simulate different replication scenarios. The Virus Agent is spawn in a random container in the grid at the beginning of the simulation (naturally multiple viruses can be spawn at the same time). Subsequently, the Virus Agent asks the cell in such container to change its own antigen peptide sequence by marking that the virus is now present in that cell. Once the infection of the cell is completed, the Virus Agent starts replicating to the neighbour cells. This is obtained by spawning a random number (according to the replication factor) of instantiations of the Virus Agent in a random set of neighbour cells. Once a new Virus Agent is created inside a neighbour cell, then the same process described above is reiterated (*i.e.*, infection and replication, in this order). Since a cell could have been previously infected by another instance of the same virus, the Virus Agent also checks if the Cell Agent can, or not, be infected. In case this is not possible, then the Virus Agent stops replicating and dies. Otherwise, it infects the Cell Agent and continues its replication.

After a certain amount of time (which can be parameterised in MAiS), the Virus Agent kills the host Cell Agent and thereby commits suicide. Hence, once the resources of a cell have been completely consumed by the virus, the virus dies (*i.e.*, the corresponding Virus Agent is killed calling the `kill()` method).

Let us now focus on the behaviours of the Virus Agent. Specifically, such agent enacts three different behaviour, which correspond to three different states of the agent.

Infecting: When an agent is born in a container, the first thing it does is to try to infect the hosting cell (*i.e.*, the Cell Agent inside the container). To achieve this in JADE, a One-Shot Behaviour is instantiated. Such behaviour has the task of communicating with the Cell Agent. In more detail, the Virus Agent informs the Cell Agent of being infected, and forcing the latter to change the antigen peptide sequence, consequently.

Cloning: After the Cell Agent inside the container has been successfully infected, the Virus Agent starts spreading amongst the neighbour cells. This is obtained by means of another One-Shot Behaviour, which calls the JADE native `createNewAgent()` method to spawn new Virus Agents inside neighbour cells. Such method is used a random number of times, on a random number of neighbour cells (depending on the replication factor).

Killing: After the cell has been infected, and the Virus Agent has successfully completed its cloning phase, the Virus Agent keeps feeding on the cell resources (simulated inside the Cell Agent). When such resources end, the Virus Agent kills the cell (by calling `kill()` method on the Cell Agent), and kills itself (by calling the `kill()` method on itself). This sequence of actions is performed inside a One-Shot Behaviour as well.

The only communication carried out by the Virus Agent is the one with the Cell Agent. The message passing involved in such communication are used to inform the Cell Agent that has been infected by a virus, and that the antigen peptide sequence needs to be updated. This is obtained by sending an `INFORM` message from the Virus Agent to the Cell Agent over the `Update_Antigen_Peptide` communication channel.

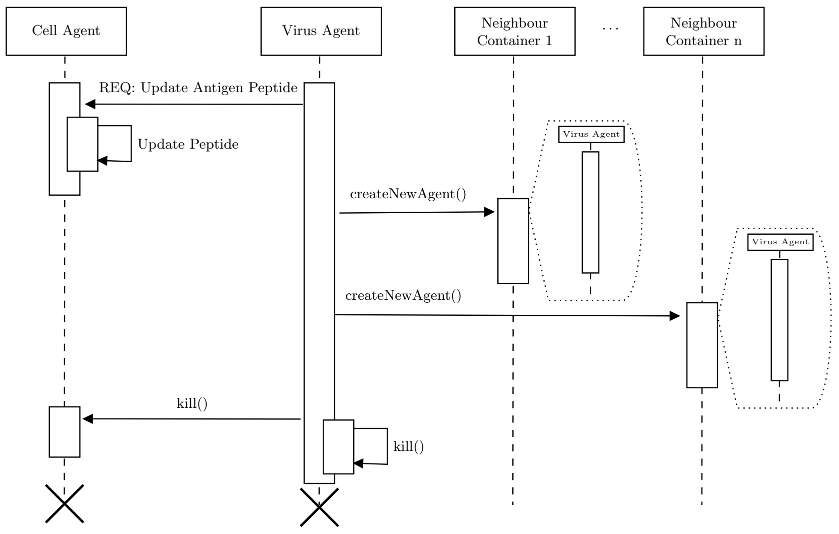

Fig. 3. Sequence diagram of the interaction amongst a Cells Agents and Virus Agents.

Figure 3 reports a Sequence Diagram of the interaction between a Cell Agent and a Virus Agent. Again, this diagram is very abstract, and serves us only to help the reader to better visualise the sequence of actions carried out by the Virus and Cell Agents. Note that, on the left, the diagram shows the actions concerning the infection of a cell. While on the right, it shows the resulting cloning phase of the virus on the neighbour containers (*i.e.*, neighbour cells, since each container contains one cell in the current definition).

5 Experiments

An initial prototype of MAiS can be found as a publicly available GitHub repository[10]. In the current state, the tool can be customised through a list of parameters. Each parameter influences a different aspect of the simulation, and can be used to model different scenarios. The available parameters are the following:

- *GRID_SIZE*: the size of the grid of cells (as shown in Figure 1, on the right).
- *T-CELL_SLEEP_TIME*: the number of milliseconds that T-Cell Agent stays in a cell before moving on with its exploration.
- *VIRUS_REPLICATION_TIME*: the number of milliseconds that a Virus Agent takes to perform replication (*i.e.*, between creation and replication).
- *VIRUS_REPLICATION_FACTOR*: the factor by which the virus multiplies.
- *CELL_IDENTIFYING_PEPTIDE*: the initial peptide of a healthy cell.
- *KILL_THE_CELL_AFTERWARD*: the number of milliseconds after which the Virus Agent kills the host cell.

Given the list of parameters presented above, we carried out experiments varying the two following parameters:

Grid Size	Given an assumption of a square-shaped universe, the grid size represents the length of a side in unit grid.
Replication factor	The replication factor of the virus (e.g., if the replication factor is n, then the virus makes n copies of itself to the adjacent containers in each cycle).

We perform some experiments with varying number of these *two* parameters while capturing the behaviour of the system as a whole. Moreover, we were interested in inspecting both the infection and death rate of the cells in the course of the simulation. Every consecutive simulation was created with an increasing number of containers (*i.e.*, cells), one Virus Agent and one T-Cell Agent. At the beginning of each simulation, the Virus Agent is dropped in a random container and starts replicating itself according to its replication factor parameter, while the T-Cell Agent, already present in the system since the beginning of the simulation, searches for the virus from one container to another. We also created a Monitor Agent to dynamically keep track of the number of infected and dead cells. Every 50 milliseconds, such agent collects the data and stores them into a *.csv* file. These log files are then used to plot the so obtained results; some of them are reported in Figure 4. Note that, every simulation is stopped when there are no virus left in the system.

[10] `https://github.com/sanchayan721/Multi_agent_Immune_System`

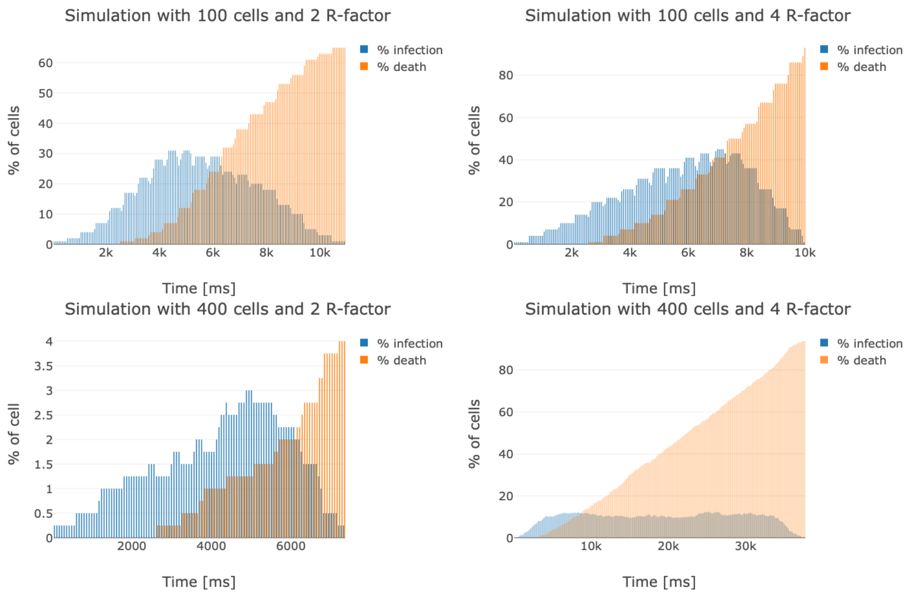

Fig. 4. Results obtained through our experiments.

Figure 4 reports four different scenarios that we have experimented. The first (top left) is a scenario with 100 cells (*i.e.*, 10 × 10 grid) and replication factor set to 2, the second (top right) with 100 cells and replication factor 4, the third (bottom left) with 400 cells (*i.e.*, 20 × 20 grid) and replication factor 2, and finally, the fourth (bottom right) with 400 cells and replication factor 4. In each scenario, we can observe how there is a delay between the percentage of infected cells, and the percentage of death cells. This derives by the intrinsic nature of infections, where a virus first infects a cell and then, after depriving the latter of all its resources, kills it. In most of the reported experiments (3 out of 4), the virus ends up killing most of the cells before being killed by the T-cell lymphocyte, or dying from starvation. Indeed, in all but the third scenario (bottom left), the percentage of death cells reaches 60% (top left), 80% (top right), and 90% (bottom right) of the total population of cells. In the third case, we have an example of simulation where the t-cell is capable of detecting the virus soon enough to stop its replication; in fact, in such simulation, the percentage of death cells reaches only 4% of the entire population. Another interesting aspect to note is the behaviour we obtain in the fourth scenario. In there, by having a large matrix (200 × 200 containers) and a high replication factor, the virus succeeds in replicating and is the scenario in fact where we obtain the highest percentage of death cells (since the fast replication is not successfully fought by the T-cell lymphocyte in such bigger area).

Before moving on with the related work section, we want to point out that MAiS is currently at the beginning of its development, and the experiments we carried out are only meant to show its initial results and to perform some empirical testing. We are already planning to build on top of MAiS, to enrich its features, and to experiment it on more challenging scenarios.

6 Related Work

Naturally, one cannot talk about agent-based simulation without citing Netlogo[11]. Indeed, also in the context of simulating the immune system, we may find many applications based upon Netlogo (thorough review on the topic [4]). When considering such applications, the most relevant difference w.r.t. MAiS is scalability. Netlogo is mainly an academic centralised simulator, and it is hard to decentralise (only features to perform concurrent executions exist), which makes it not suitable when the number of agents to simulate grows too much.

One first category of works which are related to the one presented in this paper are the ones which belong to the area of Artificial Immune Systems (AIS). In [17], the authors present an artificial immune system based intelligent multi-agent model, named AISIMAM. In [17], AISIMAM is not used to simulate how the immune system works, but as it happens in other disciplines, such as genetic programming, AISIMAM aims at exploiting mechanics which efficiently work in nature. Specifically, it applies them to a mine detection scenario. A similar work can be found in [1], where a MAS is designed to mimic the human immune system behaviour. Also here, the resulting model is applied to a certain domain of interest, which is power system reconfiguration and restoration. AIS have also been applied in the robotic scenario [6], where autonomous mobile robots emulate natural behaviours of cells and molecules to realise their group behaviours (*i.e.*, cooperation). Similarly, in [9], AIS are also used to engineer the organisational layer in a MAS, when exploited with simulated robot soccer. Another scenario where AIS have found application in MAS is the flexible job shop scheduling problem (FJSP). In [21], the authors analyse similarities between the FJSP and humoral immunity, which is one of the immune responses. Based on the similarities, a new immune multi-agent scheduling system (NIMASS) to solve the FJSP with the objective of minimizing the maximal completion time is presented. Agent-based Artificial Immune System (AbAIS) [12] is a framework which uses a hybrid architecture where heterogeneous agents evolve over a cellular automata environment and are modelled following a genetic approach. AbAIs is applied to intrusion detection systems (IDS). Finally, we can find AIS implemented in JADE as well. Also in such works, the aim is not to simulate the immune system, but to take inspiration from it to solve different tasks. We may find [16], where an AIS is developed in JADE and used in a process for gas purification from acidic components, or [8], where is used to minimize the microgrids operational cost and maximize the real-time response in grid-connected microgrids, or [10], where a mobile agent-based system architecture is proposed in JADE for machine condition monitoring by imitating human immune sys-

[11] `http://www.netlogoweb.org`

tem, or finally [5], where the AIS in JADE is used to handle the disruption management in production systems monitoring and control.

Differently from MAiS, all these works are not interested on designing, engineering and developing an agent-based simulation of the immune system; instead, they are examples of how to take inspiration from the immune system to solve general MAS problems.

The second category of works which are related to our contribution are the ones presenting agent-based simulations of the immune system. There are numerous existing works on agent-based immune system models. CAFISS [20] models cell to cell interactions in a grid with each cell denoted through a bit string. In [20], the authors describes a Java-based implementation of a framework for modeling the immune system, particularly Human Immunodeficiency Virus (or HIV) attack, using a Complex Adaptive Systems (CAS) model. The only downside of such solution is in its not being much scalable, indeed it has a large overhead caused by the use of separate threads for each cell. ImmSim [14, 3] is a simulator based on cellular automata developed in APL2 [7]; the framework exploits task parallelism on distributed computers and, because of this, it is able to reduce execution time and it supports large-scale simulations. ImmSim is also the base on which ImmunoGrid [13], and Sentinel [15], two other immune system simulators, have been built. Swarms [11] is a 3-dimensional model of the human immune system and its response to first and second viral antigen exposure (implemented in BREVE [19], a physics-based ABMS engine).

For a further readings on existing ABM techniques to simulate the immune system, a thorough review can be found in [18].

To the best of our knowledge no agent-based immune system simulation has never been proposed in JADE before.

7 Conclusions and Future Work

In this paper, we presented the building blocks of MAiS, an agent-based simulator built in JADE to reproduce the human body's immune system in a scalable and distributed manner. We are well aware that the actual immune system is extremely complex and that MAiS is only tackling a very specific aspect of it. Nonetheless, the current work allowed us to focus on the initial engineering steps required to map the immune system in JADE. In fact, we showed how some of the main organic actors (cells, lymphocyte, and so on), can be mapped into their corresponding agent representation in JADE, and how a specific step of the immune system's response can be simulated through such agents. Moreover, we showed how the innate JADE agent's mobility and containers' distributability are key features of such mapping.

As future directions, we are planning to build on top of MAiS, by adding more agents and by enriching the currently available ones. Moreover, we want to further explore the scalability and distributed features inherited by JADE, and how such features can influence the simulation process. This last aspect will be extremely valuable to perform a comparison with the other existing state-of-the-art agent-based solutions.

References

1. Belkacemi, R., Feliachi, A.: Multi-agent design for power distribution system reconfiguration based on the artificial immune system algorithm. In: International Symposium on Circuits and Systems (ISCAS 2010), May 30 - June 2, 2010, Paris, France. pp. 3461–3464. IEEE (2010). https://doi.org/10.1109/ISCAS.2010.5537841, `https://doi.org/10.1109/ISCAS.2010.5537841`
2. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology). John Wiley & Sons (2007)
3. Bernaschi, M., Castiglione, F.: Design and implementation of an immune system simulator. Comput. Biol. Medicine **31**(5), 303–331 (2001). https://doi.org/10.1016/S0010-4825(01)00011-7, `https://doi.org/10.1016/S0010-4825(01)00011-7`
4. Castiglione, F., Chiacchio, F., Pennisi, M., Russo, G., Motta, S., Pappalardo, F.: Agent-based modeling of the immune system: Netlogo, a promising framework. BioMed Research International **2014**, 907171 (2014). https://doi.org/10.1155/2014/907171, `https://doi.org/10.1155/2014/907171`
5. Darmoul, S., Pierreval, H., Gabouj, S.H.: An immune inspired multi agent system to handle disruptions in manufacturing production systems. In: International Conference on Industrial Engineering and Systems Management, IESM. vol. 2011 (2011)
6. Dioubate, M., Guanzheng, T., Toure Mohamed, L.: An artificial immune system based multi-agent model and its application to robot cooperation problem. In: 2008 7th World Congress on Intelligent Control and Automation. pp. 3033–3039 (2008). https://doi.org/10.1109/WCICA.2008.4593405
7. Falkoff, A.D.: The IBM family of APL systems. IBM Syst. J. **30**(4), 416–432 (1991). https://doi.org/10.1147/sj.304.0416, `https://doi.org/10.1147/sj.304.0416`
8. Harmouch, F.Z., Ebrahim, A.F., Esfahani, M.M., Krami, N., Hmina, N., Mohammed, O.A.: An optimal energy management system for real-time operation of multiagent-based microgrids using a t-cell algorithm. Energies **12**(15) (2019). https://doi.org/10.3390/en12153004, `https://www.mdpi.com/1996-1073/12/15/3004`
9. Hilaire, V., Koukam, A., Rodriguez, S.: An adaptative agent architecture for holonic multi-agent systems. ACM Trans. Auton. Adapt. Syst. **3**(1), 2:1–2:24 (2008). https://doi.org/10.1145/1342171.1342173, `https://doi.org/10.1145/1342171.1342173`
10. Hua, X.L., Gondal, I., Yaqub, F.: Mobile agent based artificial immune system for machine condition monitoring. In: 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA). pp. 108–113. IEEE (2013)
11. Jacob, C., Litorco, J., Lee, L.: Immunity through swarms: Agent-based simulations of the human immune system. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) Artificial Immune Systems, Third International Conference, ICARIS 2004, Catania, Sicily, Italy, September 13-16, 2004. Lecture Notes in Computer Science, vol. 3239, pp. 400–412. Springer (2004). https://doi.org/10.1007/978-3-540-30220-9_32, `https://doi.org/10.1007/978-3-540-30220-9\_32`

12. Ou, C.M., Ou, C., Wang, Y.T.: Agent-based artificial immune systems (abais) for intrusion detections: inspiration from danger theory. In: Agent and Multi-Agent Systems in Distributed Systems-Digital Economy and E-Commerce, pp. 67–94. Springer (2013)
13. Pappalardo, F., Halling-Brown, M.D., Rapin, N., Zhang, P., Alemani, D., Emerson, A., Paci, P., Duroux, P., Pennisi, M., Palladini, A., Miotto, O., Churchill, D., Rossi, E., Shepherd, A.J., Moss, D.S., Castiglione, F., Bernaschi, M., Lefranc, M., Brunak, S., Motta, S., Lollini, P., Basford, K.E., Brusic, V.: Immunogrid, an integrative environment for large-scale simulation of the immune system for vaccine discovery, design and optimization. Briefings Bioinform. **10**(3), 330–340 (2009). https://doi.org/10.1093/bib/bbp014, `https://doi.org/10.1093/bib/bbp014`
14. Puzone, R., Kohler, B., Seiden, P., Celada, F.: Immsim, a flexible model for in machina experiments on immune system responses. Future Gener. Comput. Syst. **18**(7), 961–972 (2002). https://doi.org/10.1016/S0167-739X(02)00075-4, `https://doi.org/10.1016/S0167-739X(02)00075-4`
15. Robbins, M.J., Garrett, S.M.: Evaluating theories of immunological memory using large-scale simulations. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J. (eds.) Artificial Immune Systems: 4th International Conference, ICARIS 2005, Banff, Alberta, Canada, August 14-17, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3627, pp. 193–206. Springer (2005). https://doi.org/10.1007/11536444_15, `https://doi.org/10.1007/11536444\_15`
16. Samigulina, G.A., Samigulina, Z.I.: Design of technology for prediction and control system based on artificial immune systems and the multi-agent platform JADE. In: Jezic, G., Chen-Burger, Y.J., Kusek, M., Sperka, R., Howlett, R.J., Jain, L.C. (eds.) Agents and Multi-Agent Systems: Technologies and Applications 2020, 14th KES International Conference, KES-AMSTA 2020, June 2020 Proceedings. pp. 143–153. Springer (2020). https://doi.org/10.1007/978-981-15-5764-4_13, `https://doi.org/10.1007/978-981-15-5764-4\_13`
17. Sathyanath, S., Sahin, F.: Application of artificial immune system based intelligent multi agent model to a mine detection problem. In: IEEE International Conference on Systems, Man and Cybernetics. vol. 3, pp. 6 pp. vol.3– (2002). https://doi.org/10.1109/ICSMC.2002.1176015
18. Shinde, S.B., Kurhekar, M.P.: Review of the systems biology of the immune system using agent-based models. IET Systems Biology **12**(3), 83–92 (2018)
19. Spector, L., Klein, J., Perry, C., Feinstein, M.: Emergence of collective behavior in evolving populations of flying agents. Genet. Program. Evolvable Mach. **6**(1), 111–125 (2005). https://doi.org/10.1007/s10710-005-7620-3, `https://doi.org/10.1007/s10710-005-7620-3`
20. Tay, J.C., Jhavar, A.: CAFISS: a complex adaptive framework for immune system simulation. In: Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L. (eds.) Proceedings of the 2005 ACM Symposium on Applied Computing (SAC), Santa Fe, New Mexico, USA, March 13-17, 2005. pp. 158–164. ACM (2005). https://doi.org/10.1145/1066677.1066716, `https://doi.org/10.1145/1066677.1066716`
21. Xiong, W., Fu, D.: A new immune multi-agent system for the flexible job shop scheduling problem. J. Intell. Manuf. **29**(4), 857–873 (2018). https://doi.org/10.1007/s10845-015-1137-2, `https://doi.org/10.1007/s10845-015-1137-2`