An agent-based approach to immune modelling

Dimitri Perrin, Heather J. Ruskin, John Burns, and Martin Crane

Dublin City University, School of Computing, Dublin 9, Ireland.
dperrin@computing.dcu.ie

Abstract. This study focuses on trying to understand why the range of experience with respect to HIV infection is so diverse, especially as regards to the latency period. The challenge is to determine what assumptions can be made about the nature of the experience of antigenic invasion and diversity that can be modelled, tested and argued plausibly. To investigate this, an agent-based approach is used to extract high-level behaviour which cannot be described analytically from the set of interaction rules at the cellular level. A prototype model encompasses local variation in baseline properties contributing to the individual disease experience and is included in a network which mimics the chain of lymphatic nodes. Dealing with massively multi-agent systems requires major computational efforts. However, parallelisation methods are a natural consequence and advantage of the multi-agent approach. These are implemented using the MPI library.

Keywords: HIV, immune response, complex system, agent-based, parallelisation methods

1 Introduction

The objective of this study is to understand why the range of experience with respect to HIV infection is so diverse. In particular, the work aims to address questions relating to variation in length in individual latency period. This may be very long (for relatively low success of antipathetic mutation) in one individual, compared to another with much higher mutation levels.
The indications are that the observed variation lies in the priming and initial level of fitness of the immune response of the individual, together with the various factors influencing this [1]. If such "priming patterns" can be recognised, or even predicted, then in the long term we may have a way of "typing" an individual and targeting intervention appropriately. Unfortunately, understanding how the immune system is primed by experience of antigenic invasion and diversity is non-trivial [1]. The challenge is to determine what assumptions can be made about the nature of the experience, can be modelled, tested against clinical data and hence argued plausibly. The aim is to understand how the cell interactions lead to the observed endpoints.
The immune response is dynamic and includes growth and replenishment of cells and in-built adaptability, through mutation of its defences to meet new threats.

It also includes aspects of cell mobility, which may be captured, by means of defining movement and affinity of cell-types in a defined spatial framework. In particular, this will enable study of variation in viral load and the way in which host response may lead to degradation of protection.
To investigate these questions, an "agent-based" approach is chosen, as a means of inferring high-level behaviour from a small set of interaction rules at the cellular level. Such behaviour cannot be extracted analytically from the set of rules [1], but emerges as a result of stochastic events, which play an important part in the immune response [2].
The initial model consists of functional units, called agents, with designated properties which mimic the operation of a single lymph node. This test-case prototype, however, includes all known interactions contributing to cell-mediated immunity and the local evolution of the virions. The antibody-mediated response has not been considered initially, because the cell-mediated arm plays a dominant role in repelling attack. The agents implemented represent Th (helper, or CD4) and Tc (cytotoxic, or CD8) lymphocytes, Antigen Presenting Cells, and virions. The computational structure of the numerical experiments is based on inheritance from a common C++ class designed to deal with features such as the mobility and then each class includes specific attributes and methods to implement specific properties of each cell type. The lymph node itself is modelled as a matrix, in which each element represents the physical neighbourhood of a cell type, (in terms of its agent neighbours). The frequency with which an infected cell will produce a new virion is used as the simulation time-step. At each time step, agents can move from one matrix element to another, and interact with the other agents present in their physical neighbourhood (i.e. with cell types in the same neighbourhood).
Current development is focused on increasing the number of lymph nodes, which involves millions of agents, requiring major computational effort and parallelisation methods. These are, however, a natural consequence and advantage of the multi-agent approach [3]. The aim is to extend the size and complexity of the systems modelled to something approaching realism.

2 A complex biological mechanism

2.1 The immune response against a viral attack

Immunity can be defined as all mechanisms which allow the body recognition of that which belongs to its system and consequently tolerate it, and recognise what does not and fight to eradicate it. The immune system is complex and involves various types of cells. When a foreign element is recognised, it can be dealt with in two different ways: the immune response can be non-specific or specific. A non-specific response is based upon the fact that the foreign element does not show, at its surface, the antigens characterising the cells belonging to the body. This is the response that has to be diminished when transplants are carried out. In contrast, the specific response is based on the accurate recognition of

foreign antigens. This response can be cell-mediated or antibody-mediated. The second one, also known as humoral response, is carried out by B lymphocytes and mainly targeted at bacterial attacks. We present here a few details about the cell-mediated response, targeted more specifically at viral attacks and taking place in lymphatic nodes. More details about the immune system can be found in specialised journals and immunology courses, such as [4].
The effector cell, in the cell-mediated response, is the Tc lymphocyte. However, it cannot act on its own, needing a chain reaction to achieve activation The first step is carried out by Antigen Presenting Cells which recognise foreign biological entities and start presenting these antigens at their surface. It will then encounter Th lymphocytes. If a Th cell encounters an APC presenting an antigen, which it has been specifically designed to recognise, it activates itself. The Th cells main function is then to coordinate the immune response by activating specific Tc cells.

2.2 The HIV expansion strategy

HIV virions use the Th cells described above as hosts to multiply themselves, as detailed in [5]. The gp120 glycoprotein of the virion envelope first attaches itself to the CD4 receptor, characteristic of these immune cells. Then the virion fuses with the lymphocyte using gp41 and the viral RNA is freed into the cell. The viral reverse transcriptase copies the RNA into DNA and integrates it into the cellular DNA. To be successful, this integration has to take place in activated cells. More details about this process can be found in [6]. An important aspect is the high rate of mutation: there is on average a transcription error every 10.000 nucleotides. Since the HIV genome contains about 10.000 nucleotides, this means there is on average a single difference between two "brother virions". All these mutants of course have various fates. On the one hand, most of them will result, for instance, in the suppression of an enzyme, and will be unsuccessful. On the other hand, a mutation can be successful and, for instance, modify the envelope glycoprotein, thus allowing the new virion to temporarily escape from the immune system.
The macroscopic evolution of the disease is divided into three phases. The first one corresponds to the typical immune response against a viral attack. The production of lymphocytes specific to the viral strains is launched, and within a few weeks, all the original strains are eradicated. The mutation rate here becomes critical. It has allowed the appearance of new strains, which have not been detected by the organism yet, and can therefore develop freely. As soon as a strain becomes too intrusive, its detection probability increases and it is eradicated. During this second phase, there are no symptoms. This is known as the latency period, and can last up to ten years. The immune system is heavily loaded, and the destruction of each strain also implies the destruction of the infected cell. A time comes when the immune system cannot cope with the ever increasing number of strains or remain viable, given a strong decrease of the number of the Th cells. During this last phase, known as AIDS (acquired immunodeficiency

syndrome), the whole immune system is diminished and opportunistic diseases start appearing, leading to the death of the patient.

3 Simple rules to control the agents

3.1 The agent-based approach

There is no unique definition of what an agent is. However, Wooldridge and Jennings proposed in [7] a definition which is widely accepted and specifies characteristics that an agent must have. An agent has to be autonomous: it can act without any intervention and has some control over its actions and its internal state. It has a social behaviour: it can interact with other agents thanks to a specific language. It can also react: the agent has the ability to scan part of its environment and change its behaviour to take advantage of it. The agent is proactive: it not only reacts to its environment but also acts and takes initiatives so as to satisfy goals. Building on this definition an agent-based model is a model in which the key abstraction elements are agents.
Obviously, each agent has only a limited knowledge of the world in which it evolves, and communication between agents is therefore an important aspect of this approach. This communication is sometimes referred to as linguistic actions, as opposed to non-linguistic actions which are modifications of the environment. Interaction between agents is not limited to communication: they have to share their environment. This implies that agents' actions have to be coordinated. Of course coordination does not mean cooperation: a good competitor maximizes his advantage by coordinating his actions according to the others' decisions. It also does not imply reciprocity of action: a car driver can go past another and coordinate this safely without the second driver knowing it. The key factor when choosing a coordination strategy is the size of the agent population. If every agent can interact with every other one, the number of interaction pairs increases quadratically with the population size. If interaction can occur between several agents instead of pairs, the coordination overhead increases exponentially and can easily exceed the computing facilities [8]. Developing a coordination strategy is therefore both essential and difficult. In many cases, managing to avoid conflicts and blocks is itself an important achievement. This gives us the opportunity to put the emphasis on the main drawback of this approach: it is highly resource-consuming. However, the approach also provides a solution as it is often combined with parallel methods. We develop this idea later on (section 4.2).
This approach being generic, it has been used in various fields. It has for instance been used for aerial traffic planning [9], vehicle monitoring [10] and even to manage chirurgical intensive care units [11]. It has also been extensively used in Natural Sciences, as it provides a very intuitive way to model systems: biological entities are implemented as agents, and interactions between them are dealt with through linguistic and non-linguistic actions among the agent population. In particular, the immune system itself is a discrete system in which the individual behaviour of every cell adds to create to high-level behaviour of the whole system. A simple set of local rules can therefore provide an accurate model of

this complex system. This is the approach we have chosen to take.
As we have seen earlier, most of the immune response against HIV is taking place in the lymphatic nodes. The world we model need only be a network of such nodes. The communication inside the network will be discussed later (section 4.1). Each node is implemented as a matrix. Each element of the matrix correspond to a physical neighbourhood. All the interactions between the agents therefore happen inside this local element and there is no need to consider surrounding matrix elements as would be done if using Moore or Von Neumann neighbourhoods [12].

3.2 The implemented features

There are several platforms supporting generic agent-based environments, such as Swarm [REF]. However, due to the high number of agents we plan to simulate, we think it is more efficient to have an approach fully dedicated to this particular environment, and therefore optimized. Because of the very detailled knowledge of the cell interactions, we are using a bottom-up approach: we first specify in detail the individual parts of the system (here, the agents), we then link them together to form layer componants (here, the lymphatic node), which are in turn linked until a complete system is formed (here, the lymphatic network).
As noted earlier, this study focuses on the cell-mediated response. Thus, we first need to implement three types of cells, corresponding in the code to three types of agents: Th and Tc lymphocytes, and Antigen Presenting Cells (APC). Of course, we also need a fourth type of agent to model the virions. Each type is implemented into the code using a specific C++ class.
Interestingly, even if all four types of cells have totally different roles, they have a common feature that we want to take into account, i.e. their mobility. This is implemented by another class. This class is then inherited by the four types described above. It also implements other basic properties such as the age of the agents and allow us to have the four agent classes contain only specific features; an advantage of object-oriented programming.
An agent coding a virion only has one specific attribute in the model, its viral strain. In order to prevent the code from allocating too much memory for each agent, the viral strain is only coded as an integer which links to the corresponding strain in an array containing all the useful properties of the strain (e.g. lymphocytes which recognize it, immunogenicity, etc.). The agent has a short-term and partial knowledge of its environment. It is partial in the sense that it is only knows whether there are Th cells in its physical neighbourhood (i.e. the matrix element). It is short-term in the sense that it has no memory of the evolution of the number of lymphocytes. This knowledge is the only piece of information it needs, since its unique objective is to infect a Th cell. Therefore, the typical behaviour of a virion in the model can be given as the following triptych, repeated until a lymphocyte is infected: the agent moves, scans its environment looking for a Th cell, and if possible infects the immune cell.
A Th agent has three specific attributes in the model: an integer coding its surface antigens, another integer coding its "activation state" and a third integer

coding its "infection state". Once again, it has no memory of its environment and the only part it knows of it is reduced to the presence, or not, of Tc agents. If the agent is neither activated nor infected, both integers coding the states are set to zero, and the agent's objective is only to be ready to answer an attack. There is therefore no particular action, apart from moving. The objective of an activated agent is to activate Tc cells. Its "activation state" is set to the value coding the viral strains which activated it, so that it can communicate on the threat. If the agent is infected, it produces new virions belonging to the strain coded in its "infected state", or to a new one if there is a mutation.
A Tc agent has four specific attributes: its surface antigens, its "activation state", its "expansion state" and its "memory state", all implemented as integers. The Tc agents also have a short-term and partial view of their environment: each looks only for agents having the antigens corresponding to the strain which activated it, and destroys them. When activated, an agent multiplies itself during an expansion phase, corresponding to a non-zero "expansion state". After an immune response, a small amount of the Tc agents will become memory cells: their "memory state" will keep track of the strain they fought, the reactivation will be easier, and if reactivated, the expansion phase will be more productive.
An APC agent only has one specific attribute, its "presenting state", coded as an integer. As long as the agent is not presenting any antigen at its surface, the integer stays at zero, and the agent's behaviour is focused on moving and looking for "foreign" entities in its physical neighbourhood, in order to get antigens to present. Then, the "presenting state" codes the strain corresponding to the antigens, and the agents starts looking for Th agents in order to activate them, if they are geared recognise this particular antigen.
Another aspect of the implementation chosen is the allocation of the agents. Memory allocations are among the slowest operations on a computer, and here, we have a model in which thousands of agents are created and destroyed every iteration. Dynamic allocations would make the program too slow. The approach we have chosen is to have, in each matrix element, a static allocation of the maximum number of agents we want to implement. Then, an agent moving from an element to another is coded as the transfer of its attributes from one static memory slot to another. Every agent being small and with few attributes, this gives satisfying results.

3.3 How to deal with stochastic events?

In this model, most methods and functions have to include random number generation. This is due to the fact that many aspects of the real-life system involve stochastic events. More details can be found in [2], but here are a few examples. First, an aspect we have to deal with is the process by which new lymphocytes are created. A lymphocyte can only recognize a specific set of antigens so, to protect itself against any attack, the body has to generate thousands of "variations" between lymphocytes. This has to be implemented using random numbers. Likewise, we noted that one of the most decisive features of the virions is their high

mutation rate, and this implies another use of random numbers. Finally, there is no sensible way to deal with mobility unless we include stochasticity.
Stochastic events are essential to this work and a reliable random number generator is needed. A full-scale model will involve millions of agents in very long simulations. Therefore, the generator also has to be very efficient. As parallel aspects are involved, it would also be a plus for the generator to include such features. There are many generators available, and good ones can also be designed explicitely (see e.g. [13]). However, due to our model requirements, what is needed here is a top-quality parallel generator, and we chose to use the Scalable Parallel Random Number Generators library (SPRNG) [14]. This library incorporates recent, state-of-the-art, developments in the mathematics and computer science of parallel pseudorandom number generation. It an efficient library with an existing, active, user base, ensuring high standards. It allows the streams to be also absolutely reproduced, for computational verification, independent of the number of processors used in the computation and of the loading produced by sharing of the parallel computer. Using it, we can be confident we will produce statistically significant results at a very low computing cost.

4 Interactions between the lymphatic nodes

4.1 Sharing knowledge and transferring agents

The immune system is organised so that every lymphatic node is a small defence unit in which the immune response is taking place. There is no need for the response to take place in every node, which is why we built our model as a network of independent matrices (putting the emphasis on the local model of the node). The only physical exchange between lymphatic nodes happens through the recirculation and the mobility of cells which go from one node to another. Each node in the model therefore needs an entry point and an exit point. If, when moving inside the node, an agent reaches the exit point, it is removed from the node and put into a transfer list. The list is dealt with at the end of the iteration. In the meantime, other agents move, interactions take place, as time passes. This accounts for the time it takes the agent in real-life to commute between two nodes. The way in which agents are transferred between the nodes mimics the transfer between matrix elements: we consider only attributes, rather than the agent itself. Thus, an entry in the transfer list contains the type of the agent, its attributes, and its destination. At the end of the iteration, all lists are put together and the moving agents are transferred to the entry point of their destination node.
The other aspect of the communication between our nodes is inherent to our implementation. Since we decided not to put all the strain properties into each agent, we need a way to code them somewhere and make them available to all the agents, wherever they are in the model. These are important properties, and must not be neglected. For instance we need to know, for each strain, which lymphocytes will recognise it for sure and which lymphocytes might recognise it. One characteristic is that when a lymphocyte from the second category recognise

the strain, it moves from the second into the first. This is critical to the realism of the model, since it allows us to introduce some adaptability and emergent behaviour. One answer could have been to create a linked list containing the strains active in the current simulation. The obvious advantage is to limit the size allocated to the strains to what is actually needed. However, it has one major drawback which makes it pointless in our case, namely that the high mutation rate means a large number of strains, increasing as the simulation continues. The bigger the list, the longer it will take to get the properties for a particular strain and since this list has to be accessed thousands of times in every iteration, this process would slow the whole program down. We therefore decided to have an array of strains. This array is large (i.e. tens of thousands of strains) and represents potential strains for the simulation to be implemented. Considering that a strain in the array can account for various strains in real life (if they differ on properties we do not code explicitely), we are confident this should give us enough diversity.

4.2 Parallelisation efforts

When the program is running at full scale, each node contains hundreds of thousands of agents. In real life, a human body contains about a thousand lymphatic nodes. Matching this value is a long-term objective and may not be achievable, but even with fifty nodes, we would have to deal with millions of agents. The time-step of the program is about fifty seconds, so about six million iterations are needed for a 10-year simulation. Running such a program on a single computer would take months, and not even have enough memory might be available to initialize all the matrices. If we also consider the fact that we have to run several simulations to statistically assess the role of each parameter such as the mutation rate, a parallel approach makes even more sense.

The approach we develop here is to mimic the immune system, in the sense that each lymphatic node will be computed by a different computer (also called node) on a cluster. As the lymphatic nodes are mainly independent from each other, this is the best way to take advantage of the parallel option. Moreover, the local model is already known to run on a single computer so approximate expectations on performances are known also. This type of spatial parallelisation has been studied in [15] for Monte-Carlo simulations. The main disadvantage in that study is the communication overload. Here, most of the communication taking place on the cluster is the transfer of agents from one node to another. Using the list process described above, this is kept to a minimum. This parallel approach is implemented using the Message-Passing Interface (MPI) [16, 17]. It is under validation on a cluster composed of a Dell PowerEdge 1750 acting as the master node and sixteen of these machines acting as slaves. More important clusters will also be used for full-scale runs.

The most difficult part here is to deal with the updates of the array containing the strain. On the one hand, if we keep only one array (on the main node of the cluster) it would lead to excessive communication: each agent would have to ask for the viral strain properties at each iteration. On the other hand, having an

array linked to every node would impose a process to make sure that at every instant all arrays contain the same information, for all the strains. Using MPI advanced features, this can be done through "collective communication".
This approach provides an intuitive way to combine the parallel computing features with a process which mimics the immune system. The transfer of the agents is currently being optimized. This will allow us to then run full-scale simulations. Our objective is to first reproduce the three-phase evolution of the disease and then alter the parameters (mobility, viral load) to study how they affect the latency period length.

5 Conclusion

The objective of this study is to understand why the range of experience with respect to HIV infection is so diverse, addressing in particular questions relating to variation in length in individual latency period. To investigate these questions, an "agent-based" approach is chosen, as a means of inferring high-level behaviour from a small set of interaction rules at the cellular level as well as including stochastic events.
The model developed mimic the immune system, as it is organised as a network of matrices, each of them corresponding to a lymphatic node. Matrix elements can host several agents, of four different types, accounting for virions, Th and Tc lymphocytes, and Antigen Presenting Cells. Thus, it is possible to model the HIV spreading strategy and the cell-mediated immune response.
Because the system we study is so complex, millions of agents are needed, and it is not possible to run the model on a single computer. Therefore, parallel methods are implemented. Using MPI, every lymphatic node is allocated to a different computer on a cluster, and "collective communication" is used to share knowledge common to all nodes.
This parallel implementation is currently being tested and the first results should be available in the coming months.

6 Acknowledgements

The authors would like to thank the Irish Research Council for Science, Engineering and Technology for the funding made available through the Embark Initiative.

References

1. Burns, J.: Emergent networks in immune system shape space. PhD thesis, Dublin City University, School of Computing, 2005.
2. Germain, R.N.: The Art of the Probable: System Control in the Adaptive Immune System. Science 239 **5528** (2001) 240–245.
3. Jennings, N., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous agents and multi-agents systems 1 **1** (1998) 7–38.

4. Lemahieu, J.C..: Le systeme immunitaire. Immunology courses [French] (available online at http://anne.decoster.free.fr/immuno/orgcelri/orgcelmo.htm), last access on December 14th, 2005.
5. Klatzmann, D., Champagne, E., Chamaret, S., Gruest, J., Guetard, D., Hercend, T., Gluckman, J.C., Montagnier, L.: T-lymphocyte T4 molecule behaves as the receptor for human retrovirus LAV. Nature 312 **5596** (1984) 767–768.
6. Decoster, A., Lemahieu, J.C..: Les retrovirus. Immunology courses [French] (available online at http://anne.decoster.free.fr/d1viro/vretrov0.html), last access on December 14th, 2005.
7. Wooldridge, M., Jennings, N.: Intelligent agents: Theory and practice. The Knowledge Engineering Review 2 **10** (1995) 115–152.
8. Durfee, E.H.: Scaling up agent coordination strategies. Computer 34 **7** (2001) 39–46.
9. Cammarata, S., McArthur, D., Steeb, R.: Strategies of cooperation in distributed problem solving. roceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), Karlsruhe, Germany (1983).
10. Durfee, E.H.: Coordination of distributed problem solvers. Kluwer Academic Publishers (1998).
11. Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R., Seiver, A.: Distributing intelligence within an individual. Distributed Artificial Intelligence Volume II, L. Gasser and M. Huhns (editors), Pitman Publishing and Morgan Kaufmann (1989) 385–412.
12. Kari, J.: Theory of cellular automata: A survey. Theoretical Computer Science 334 **2005** (2005) 3–35.
13. Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P.: Numerical Recipes in C++: the art of scientific computing. Cambridge University Press (2002).
14. Srinivasan, A., Mascagni, M., Ceperley, D.: Testing parallel random number generators. Parallel Computing 29 **2003** (2003) 69-–94.
15. Hecquet, D., Ruskin, H.J., Crane, M.: Optimisation and parallelisation strategies for Monte Carlo simulation of HIV infection. Submitted to Computers in Biology and Medicine (2005).
16. Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming With the Message-Passing Interface, second edition. MIT Press (1999).
17. Gropp, W., Lusk, E., Skjellum, A.: Using MPI-2: Advanced Features of the Message Passing Interface. MIT Press (1999).