World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:2, No:4, 2008

HIV Modelling - Parallel Implementation Strategies

Dimitri Perrin, Heather J. Ruskin, and Martin Crane

***Abstract*—**We report on the development of a model to understand why the range of experience with respect to HIV infection is so diverse, especially with respect to the latency period. 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 network of independent matrices mimics the chain of lymph nodes. Dealing with massively multi-agent systems requires major computational effort. However, parallelisation methods are a natural consequence and advantage of the multi-agent approach and, using the MPI library, are here implemented, tested and optimized. Our current focus is on the various implementations of the data transfer across the network. Three communications strategies are proposed and tested, showing that the most efficient approach is communication based on the natural lymph-network connectivity.

***Keywords*—**HIV, Immune modelling, MPI, Parallelisation.

I. 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 rules governing movement and affinity of cell-types in a defined spatial framework. In particular, this enables 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 cellmediated arm plays a dominant role in repelling the 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 inclusion of 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 timestep.

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). The implementation of the neighbourhood will be discussed in section III-A.

Lymph nodes involve millions of agents and require major computational effort and parallelisation methods. These are, however, a natural complement to the multi-agent approach [3]. Our current objective is to implement an efficient data transfer across our network of nodes, in order to facilitate the long-term aim to extend the size and complexity of the systems modelled to something approaching realism.

Manuscript received August 28, 2006. This work was supported in part by the Irish Research Council for Science, Engineering and Technology (Embark Initiative).

All authors are with Dublin City University, School of Computing.
Dimitri Perrin (corresponding author, phone: +353-1-700-8449; fax: +353-1-700-5442; e-mail: dperrin@computing.dcu.ie).

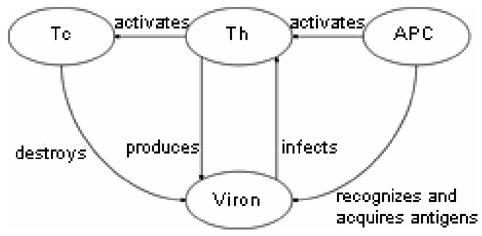

Fig. 1 Cell-level interactions

II. The Biological Background

A. The Immune Response against a Viral Attack

Immunity can be defined as a function of all mechanisms which permit the body to recognise entities belonging to its system (which consequently it tolerates), and those that do not (which it fights). 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 e.g. 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 latter one, also known as humoral response, is carried out by B lymphocytes and mainly targeted at bacterial attacks. The cell-mediated response is targeted more specifically at viral attacks and takes place in lymph nodes. Brief details follow. Full discussion on the immune system can be found in specialised journals, texts and web-based materials see e.g. [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. These 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. These cell interactions are shown in Fig. 1.

B. The Lymph Network

When facing a viral attack, the most significant part of the cell-mediated response is located within small organs called lymph nodes. These organs are distributed throughout the body and, in humans, number about a thousand, which act as small defence units. These are thus loci for Tc lymphocytes activation, multiplication and attack on the virions. To provide an efficient scanning and filtering mechanism for the body, the lymph nodes are linked through a network. The cell mobility along that network is expected to have a strong influence on the immune response, and modelling it is, therefore, an important objective of our study.

C. The HIV Expansion Strategy

HIV virions use the Th cells, described above, as hosts to multiply themselves, as detailed in [5] and shown in Fig. 1.

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. (A detailed description of this process can be found in [6]. An important aspect is the high rate of mutation: on average there is a transcription error every 10,000 nucleotides. Since the HIV genome contains about 10,000 nucleotides, this implies a single difference on average between two “brother virions”. Most mutations result, for instance, in the suppression of an enzyme, and are unsuccessful. On the other hand, a successful mutation may e.g. modify the envelope glycoprotein, thus allowing the new virion to temporarily escape the immune system defences.

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 is critical. It can facilitate 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 the large decrease in 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.

III. The Modelling Strategy

A. 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 opponents' decisions. It also does not imply reciprocity of action: a car driver can go past another and coordinate this safely without involvement of the second driver. 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 computing facilities available [8]. Developing a coordination strategy is therefore both essential and difficult. In many cases, managing to avoid conflicts and blocks is itself a significant achievement.

The approach is generic, and has been used in various fields, including aerial traffic planning [9], vehicle monitoring [10] and even management of chirurgical intensive care units [11]. It has also been extensively used in the 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 is aggregated to create 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.

With respect to the immune response to HIV, most activity (as noted earlier) takes place in the lymph nodes. The world is thus a network of such nodes. The communication inside the network will be discussed later, (section III-C). Each node is implemented as a matrix. Each element of the matrix corresponds 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, which would be required if using Moore or Von Neumann neighbourhoods [12].

B. The Implemented Features

There are several platforms supporting generic agent-based environments, such as Swarm [13]. However, due to the high number of agents in the system modelled, it is more efficient to have an approach fully dedicated to our particular environment, and therefore optimized. The very detailed knowledge of the cell interactions dictates a bottom-up approach: we first specify in detail the individual parts of the system (the agents), then link them together to form layer components (the lymph node), which, in turn, are linked until a complete system is formed (the lymph network).

In focusing on cell-mediated response, we need to implement three types of host cells, corresponding in the code to three types of agents: Th and Tc lymphocytes, and Antigen Presenting Cells (APC). A fourth type of agent is used to model the virions. Each type is implemented into the code using a specific C++ class. Despite cell types having totally different roles, the common feature to take into account is their mobility. This is implemented by an additional class, inherited by the four types described above. It also implements other basic properties such as the age of the agents and permits the four agent classes to contain only specific features; (an advantage of object-oriented programming).

An agent coding a virion has only one specific attribute in our model, namely its viral strain. In order to prevent the code from allocating too much memory for each agent, the viral strain is 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 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".

- If the agent is neither activated nor infected, its objective is only to be ready to respond to 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. When activated, an agent multiplies itself during an expansion phase, corresponding to a non-zero "expansion state". After primary 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 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 start looking for appropriate Th agents in order to activate them, if primed to recognise this particular antigen.

Another aspect of the implementation presented is the allocation of the agents. Memory allocations are among the slowest operations on a computer. 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 chosen is to have, in each matrix element, a set of integers, one for each potential agent located there. An array, for which the size is fixed to the maximum number of agents we want to implement, is statically allocated, and each integer represents an offset used to find the agent in the array. Then, an agent moving from one element to another is coded as the alteration of only two integers, one in each element, and the creation/destruction of an agent alters only one local attribute.

C. Interactions between the Lymph Nodes

The immune system is organised so that every lymph node is a small defense unit, which mounts a unilateral immune response. Not all nodes need to be engaged in the response at any one time; our model is thus a network of independent matrices; (with the emphasis on the local model of the node).

The only physical exchange between lymph 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 and interactions take place over time (equivalent to the time taken for 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.

IV. Parallelisation Efforts

A. What kind of Parallelisation?

When the program is running at full scale, each node contains hundreds of thousands of agents. Matching the real-body count of a thousand lymph nodes is a long-term objective and may not be achievable, but even for fifty nodes, we 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 the memory needed to initialize all the matrices might not even be available. 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, the need for a parallel approach is clear.

The immune system is mimicked in our model by permitting each lymph node experience to be computed by a different computer (called computing node) on a cluster. As the lymph nodes are effectively 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 also known. This type of spatial parallelisation has been studied, for Monte-Carlo simulations [14], with the main disadvantage being 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, we keep this to a minimum. This parallel approach is implemented using the Message-Passing Interface (MPI) [15], [16]. It was validated on a cluster composed of a Dell PowerEdge 1750 acting as the master node and sixteen of these machines acting as slaves. Larger clusters will also be used for full-scale runs.

B. A List to Transfer the Agents

Even when kept to a minimum, communication between computing nodes is always a bottle-neck on this type of model. As the system size increases, a bad communication strategy could have devastating effects on the computation time; e.g stochastic aspects of our model, such as mutations, require several simulations for each set of parameter, and cannot afford inefficiency. The aim is to transfer information optimally about the agents leaving the nodes.

A first solution is to have one single list, containing the agents' attributes and their type. This leads to a list containing blocks of eight integers, one block for each agent. For most agents, i.e. all but those coding Tc cells, a part of the block will stay empty. A further solution is to have a different list for each type of agent. Since the need to specify the agent type is eliminated, and since the number of attributes of each agent in the list is now fixed, the block size is now seven for the Tc list, six for the Th list, and only four for the virion and APC lists. However, this solution also implies sending information four times as often as for a single list, and the "latency" of the physical network may result in a slower communication.

These two solutions were tested on the cluster described above, for various numbers of nodes and agents. It appears that, as the number of nodes increases, it becomes more efficient to use a single list. This is explained by the network latency and the way MPI works. Before sending anything, the sender and the receiver must both know the size of what is transferred. Thus, when a list has to be sent, the first step is to send the size of the list (always an integer). Thus for our implementation, an integer i is sent, and, for i 6= 0, a list of integers follows. If the list is in fact empty, some time is wasted due to the latency.

With only one list, an empty one is unusual, but with four, it becomes a regular feature of the iterations. For instance, a non-infected lymph node will always send a zero for the size of its virion list. The more nodes we have, the more often this happens, and the gain in the amount of transferred data is outweighed by the wasted time. For this reason, we have opted for a single list.

A further consideration is the frequency of sending lists across the network. More efficient communication implies sending non-empty lists. Obviously, the longer we wait before sending a list, the bigger this list gets. In Table I, computation times are shown for 20,000 iterations, when we send at the end of every or every other time-step. The program appears slightly faster when we communicate data less often. However the gain is not significant for very low agent count: the few agents are scattered in the lymph node and are less likely to reach the exit point. The improvement is highest for medium agent count: for a high count, it is likely that at least one agent will reach the exit point, and iterations leading to an empty list are less common, but do occur. We observe an improvement when sending only every other iteration; this pattern is confirmed if we wait three, four, or five iterations before sending the lists. There are, however, two limitations.

The first is a memory concern, since an ever bigger list is resource-consuming. More importantly, there are biological considerations involved. A time-step is equivalent to fifty seconds, and the number of iterations must therefore be kept close to the actual time estimated for a cell to commute from one node to another. Separating the communication phases by more than five iterations is thus less realistic and should be avoided.

TABLE I
INFLUENCE OF THE FREQUENCY AT WHICH THE LISTS ARE SENT - COMPUTATION TIME FOR 20,000 ITERATIONS AND 16 NODES

Configuration	Communication every iteration	Communication every other iteration
Low agent count	377 sec.	-1.48%
Medium agent count	982 sec.	-34.9%
High agent count	2187 sec.	-10.8%

C. *Different Implementations of the Lymph Network*

The final part of the implementation seeks to optimize the sending method. There are many different solutions: we focus on three and their potential improvements.

1) Every agent can “physically” go from a given node to any other, with a function in the model deciding where each individual agent will actually go. Since every node can send agents to any other one, one solution might be for each to share its list with all other nodes at the same time. Using MPI, this is made possible by the broadcast function (MPI Bcast). On a 16-node simulation, a communication phase will start with sixteen successive broadcasts. Then, each node has the list of all the agents leaving any node, so we need only look through this to find those arriving at the current node. The main drawback of this approach is that destination nodes receive more data than they actually need, since they receive information about all the agents which left their host node.

2) To avoid unnecessary data transfer, we cannot use broadcast and must opt for direct communication. However, direct transfer between every couple of nodes would, on many occasions, lead to sending information about an empty list, thus slowing the program down, (as found for the four-list solution). For this approach to be efficient, we need a third node to act as the middle-man, with all the nodes sending their list to this one. Here, the agents are sorted according to their destination, and, to every node, a list is sent, containing only the agents which are relevant. The main drawback for this one is that a node can only receive from (or send to) one other node at a time. It implies that in the meantime, the others are idle.

a) Dedicating one node on the cluster only to this role of middle-man ensures it is always ready to send and receive, rather than in the middle of iteration.

b) Inclusion of an iteration between the sending of the first list, (agents leaving a node) and the reception of the second list, (agents arriving at that same node), prevents “computing nodes” from being idle, and gives time for the “middle-man node” to finish receiving every list and sorting the agents.

c) Creation of subnetworks. As the number of nodes increases, so does the time one given node has to wait before being able to send/receive. An alternative is to create more “middle-man nodes”. On a 16-node cluster, we could have four groups, each formed with three “computing nodes” to deal with modelling and one used for communication. The first three would run iteration, send their list, compute iteration, and receive the new list. The last one receives the lists, shares information with the other similar nodes, and sends the new lists. With this configuration, any node has a maximum of three nodes before it in the queue, and the program is expected to be faster as a result.

3) The last type of communication is a transcription of the real lymph network. If it is true that the lymph network is connected (in the graph theory context), it does not imply that it is complete, and in fact it is not: if we take two lymph nodes, it is likely that there will be no direct connections between them (incomplete), even though there is always a path from one to the other (connected). These properties can be used to implement the lymph network. A network can be created explicitly, rather than by a function as described above; communication can be physically limited to this network. Without creating any biological issues, we can also impose the requirement that nodes have either two (one incoming and one outgoing) or three connections (two incoming and one outgoing, or vice versa). This would imply that for any given node, at any stage of the simulation, there is a maximum of two nodes in front in the queue.

a) The network can also be designed to satisfy two-colouring only, thus decreasing the communication load: during odd iterations, black nodes send data and white ones receive it, and vice versa during even iterations.

These approaches, shown in Fig. 2, were tested, and results are shown in Table II. The broadcast approach is clearly to be avoided due to its inefficiency. It gave useful results only on very small networks, (four nodes), whereas our aim is to have as many nodes as possible. The subnetwork approach was tested on 16 nodes, but since only 12 of them are dealing with modelling the lymph units, the results are meaningful only if compared to cases running with 12 lymph nodes. In such

TABLE II
COMMUNICATION STRATEGIES - COMPUTATION TIME COMPARED TO THAT OF THE SIMPLE DIRECT COMMUNICATION, FOR 20,000 ITERATIONS AND 16 NODES

Communication strategy	Computation time
1	3.18
2	1.00
2.a	0.97
2.b	0.92
2.c	N/A
3	0.90
3.a	0.83

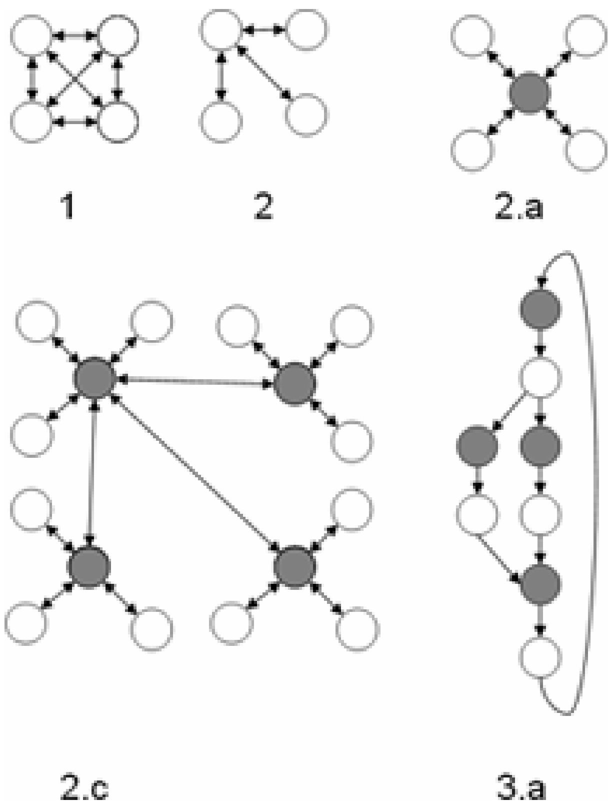

Fig. 2 Communications strategies - On 2.a and 2.c, grey nodes are dedicated to communication

comparisons, the subnetworks do not offer any significant improvements. For the other approaches, the results confirm expectations: the most efficient communication strategy is the natural one i.e. that which mimics the lymph network in its structure and uses colouring as a way to optimize the data transfer. The overall improvement is slightly under 20%, but as communication is only one aspect of the model, it is still an important gain. For a full ten-year simulation, representing millions of iterations, this saves hours of computation.

V. CONCLUSION

The objective application 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 including stochastic events.

Due to the size and complexity of the model, parallels methods are implemented, using MPI. Every lymph node is coded as an independent matrix and allocated to a different computer on a 16-processor cluster. Our current focus is on optimization of the data transfer across the network of matrices.

Three strategies were proposed, along with several ways to improve them. Tests run on the local cluster showed that the most efficient approach is to create a network between the lymph nodes, similar to that found in a body, and to colour this network so as to balance the data transfer between the nodes. Full-length simulations are now starting on the smallscale cluster, before moving to a more powerful one of 64 computing nodes.

REFERENCES

[1] J. Burns. Emergent networks in immune system shape space. PhD thesis, Dublin City University, School of Computing, 2005.
[2] R.N. Germain. The art of the probable: System control in the adaptive immune system. Science, 293(5528):240–245, 2001.
[3] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. Autonomous agents and multi-agents systems, 1(1):7–38, 1998.
[4] J.C. Lemahieu. Le syst`eme immunitaire. Immunology courses [French] (available online at http://anne.decoster.free.fr/immuno/orgcelri/orgcelmo.htm), accessed on December 14th, 2005.
[5] D. Klatzmann, E. Champagne, S. Chamaret, J. Gruest, D. Guetard, T. Hercend, J.C. Gluckman, and L. Montagnier. T-lymphocyte T4 molecule behaves as the receptor for human retrovirus LAV. Nature, 312(5996):767–768, 1984.
[6] A. Decoster and J.C. Lemahieu. Les r´etrovirus. Virology courses [French] (available online at http://anne.decoster.free.fr/d1viro/vretrov0.html), accessed on December 14th, 2005.
[7] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 2(10):115–152, 1995.
[8] E.H. Durfee. Scaling up agent coordination strategies. Computer, 34(7):39–46, 2001.
[9] S. Cammarata, D. McArthur, and R. Steeb. Strategies of cooperation in distributed problem solving. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), Karlsruhe, Germany, 1983.
[10] E.H. Durfee. Coordination of distributed problem solvers. Kluwer Academic Publishers, 1998.
[11] B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett, and A. Seiver. Distributing intelligence within an individual. In L. Gasser and M. Huhns, editors, Distributed Artificial Intelligence Volume II, pages 385–412. Pitman Publishing and Morgan Kaufmann, 1989.
[12] J. Kari. Theory of cellular automata: A survey. Theoretical Computer Science, 334(2005):3–35, 2005.
[13] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system: A toolkit for building multi-agent simulations. Working Paper 96-06-042, Santa Fe Institute, 1996.
[14] D. Hecquet, H.J. Ruskin, and M. Crane. Optimisation and parallelisation strategies for monte carlo simulation of HIV infection. To appear in Computers in Biology and Medicine, 2006.
[15] W. Gropp, E. Lusk, and A. Skjellum. Using MPI: Portable Parallel Programming With the Message-Passing Interface, second edition. MIT Press, 1999.
[16] W. Gropp, E. Lusk, and A. Skjellum. Using MPI-2: Advanced Features of the Message Passing Interface. MIT Press, 1999.