ADAPTIVE OPTIMISTIC SYNCHRONISATION FOR MULTI-AGENT DISTRIBUTED SIMULATION

MICHAEL LEES
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
mhl@cs.nott.ac.uk

BRIAN LOGAN
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
bsl@cs.nott.ac.uk

GEORGIOS THEODOROPOULOS
School of Computer Science
University of Birmingham
Birmingham B15 2TT, UK
gkt@cs.bham.ac.uk

ABSTRACT In this paper we describe an adaptive, optimistic synchronisation mechanism for the parallel discrete event simulation of agent-based systems. The mechanism uses the Sphere of Influence (SoI) of an event (the region of the shared simulation state read or written to by the event) to define an adaptive metric which can be used with a throttling mechanism such as moving time windows. We show how such a metric can be calculated by monitoring the common reads and writes made by the agents to the shared simulation state modelling the agent's environment, and present the results of our preliminary investigations into the relationships between agent read and write patterns and rollback frequency.

Keywords : Distributed Simulation, Agent-based Systems, Synchronisation.

1. INTRODUCTION

An *agent* can be viewed as a self-contained, concurrently executing thread of control that encapsulates some state and communicates with its environment (in which the agent is embedded) and possibly other agents via some sort of message passing. The *environment* of an agent is that part of the world or computational system 'inhabited' by the agent. The environment may contain other agents whose environments are disjoint with or only partially overlap with the environment of a given agent. Agent-based systems offer advantages when independently developed components must inter-operate in a heterogeneous environment and are increasingly being applied in a wide range of areas including telecommunications, business process modelling, computer games, control of mobile robots and military simulations.

While agent-based systems offer great promise, their adoption has been hampered by the limitations of current development tools and methodologies. Multi-agent systems are often extremely complex and it can be difficult to formally verify their properties. As a result, design and implementation remains largely experimental, and experimental approaches are likely to remain important for the foreseeable future. In this context, simulation has a key role to play in the development of agent-based systems, allowing the agent designer to learn more about the behaviour of a system or to investigate the implications of alternative agent architectures, and the agent researcher to probe the relationships between agent architectures, environments and behaviour.

In [Logan and Theodoropoulos, 2001] a parallel discrete event simulation framework for multi-agent systems is presented. Identifying the efficient distribution of the agents' environment (namely, the *shared state*) as a key problem in the simulation of agent-based systems, the framework models agents as Logical Processes and the environment as a tree-shaped network of processes (referred to as *Communication Logical Processes* or *CLPs*) which is dynamically reconfigured to reflect the changing interaction patterns between the agents and their environment in the simulation.

The central concept of the framework is the notion of the *Sphere of Influence* (SoI). The SoI of an event is defined as the set of state variables read or written as a consequence of the event and depends on the type of event (e.g., sensor events or motion events), the state of the agent or environment logical process which generated the event and the state of the environment. The SoI of an event is limited to the immediate consequences of the event rather than its ultimate effects, which depend both on the current configuration of the environment and the (autonomous) actions of other agents in response to the event. The SoI of an agent process p_i over the time interval $[t_1, t_2]$, $s(p_i)$, is then defined as the union of the spheres of influence of the events generated by the process over the interval. In [Logan and Theodoropoulos, 2001], the SoI of the LPs are used to derive an idealised decomposition of the shared state into CLPs to facilitate dynamic load balancing and interest management. In this paper, we discuss how the SoI can be exploited in the design of an adaptive synchronisation mechanism.

The rest of the paper is organised as follows: In sections 2 and 3 we give a brief introduction to simulation and synchronisation and discuss the role of the shared state in an agent simulation. In section 4 we describe our adaptive synchronisation mechanism and present a metric based on Spheres of Influence. Although the metric could be used with any optimism limiting mechanism, for clarity and ease of explanation we assume a window based scheme where a smaller window implies lower optimism (e.g., moving time windows [Sokol and Stucky, 1990]). In section 5 we present our experimental results. The paper concludes with section 6 where we touch on possible future work.

2. SYNCHRONISATION

Every simulation model specifies the physical (real) system in terms of *events* and *states*. Executing a simulation therefore consists of 'processing' events, which correspond to real events in the physical system. Simulations can be classified

into two groups depending on the way events are processed and state updates occur: *continuous* and *discrete*. In a continuous simulation, state changes occur continuously, whereas in a discrete simulation events occur at fixed points in time and execute instantaneously. In an *event driven* simulation state variables are updated only when something interesting occurs, i.e., an *event*. Each event occurs at a particular instant in simulation time and the event has this time associated with it, this is known as the *time-stamp* of the event. A single processor (sequential) discrete event simulation consists of the following,

- *State Variables* – collectively describe the state of the system
- *Event list* – list of events to be processed
- *Global clock* – denotes the simulation time

If the discrete event simulation is split into multiple *Logical Processes* (LPs) and spread across multiple machines it becomes a *Parallel* Discrete Event Simulation (PDES).

A sequential discrete event simulation can easily ensure that events are processed in time stamp order as it processes the event with the smallest time stamp in the event list. Spreading the simulation over multiple processes (PDES) requires multiple event lists, one for each LP. A consequence of this is that ensuring the events are processed in time stamp order is less straightforward. In asynchronous, event-driven distributed simulation, each LP maintains its own local clock with the current value of the simulated time, Local Virtual Time (LVT). This value represents the process's local view of the global simulated time and denotes how far in simulated time the corresponding process has progressed. With each LP processing its event list independently and advancing its LVT at its own rate, it may be the case that events are processed out of time stamp order. Therefore a mechanism is required to ensure the parallel simulation faithfully implements the causal dependencies and partial ordering of events dictated by the causality principle in the modelled system.

It has been shown [Lamport, 1978] that a distributed system consisting of asynchronous concurrent processes will not violate the causality principle if each process consumes and processes event messages in non-decreasing timestamp order (the *local causality constraint* (LCC)). There are two main approaches to ensuring that the local causality constraint is not violated: *conservative* and *optimistic*. Conservative mechanisms strictly avoid violation of the LCC while optimistic mechanisms provide a means to undo computation which causes a violation. In more recent years hybrid mechanisms which take aspects of both have been developed, i.e., optimistic schemes with constrained optimism such as moving time window [Sokol and Stucky, 1990]. Other optimistic schemes have been developed so that the degree of optimism (how constrained they are) can be decided at run time. These are known as adaptive synchronisation mechanisms (e.g. [Ferscha, 1995]).

3. SHARED STATE AND SPHERES OF INFLUENCE

Consider an agent simulation with two agents, A1 and A2. The shared state of the their environment can be modelled as a table (see Table 1). The table shows the *read* and *write* patterns for each variable. We use the term *access* to indicate either a read or a write.

Variable	Access patterns
x_1	$(A_2, R, t = 1), (A_2, R, t = 3), (A_1, W, t = 2)$
x_2	$(A_1, R, t = 2), \ldots$
.	
.	
.	
x_n	$\ldots$

Table 1: A global view of the shared state

Table 1 depicts the access patterns for two variables in the shared state. Each access is represented by a triple: A_n represents the agent performing the access, R/W represents whether the access was a read or write and t represents the virtual time at which the access occurred. The left to right ordering indicates when the access arrived in real time. So the access $(A_2, R, t = 1)$ arrived before $(A_2, R, t = 3)$ for variable x_1.

The table also depicts a *rollback pattern* occurring on variable x_1. A *rollback* pair consists of a read R made by LP_i with time stamp T_R and a write W made by LP_j with time stamp T_W $(j \neq i)$. We say a rollback pair is a rollback pattern on variable x if $T_W < T_R$. A rollback pattern results in an actual rollback when the write W is realised after the read R in real time. For variable x_1 in Table 1 this occurs when A_1 performs the write at t=2. The read performed by A_2 with virtual time stamp t=3 arrived, in real time, before the write performed by A_1. This means the read performed by A_2 won't reflect the correct value and so A_2 needs to rollback to before the read occurred. In terms of synchronisation a key observation is that the probability of rollback is increased when many different LPs read and write the same variables. We now extend and clarify the definition of SoI from [Logan and Theodoropoulos, 2001] by splitting the SoI into two distinct sets:

1. The *sphere of influence of Writes* (SoI_W), which contains the set of variables written to by the LP over the time period $[t_1, t_2]$.
2. The *sphere of influence of Reads* (SoI_R), which contains the set of variables read by the LP over the time period $[t_1, t_2]$.

Considering the example given above we can now say:

1. Any variable which appears only in SoI_W for all LPs (agents) over $[t_1, t_2]$ (i.e., no agent reads the variable) is not important in terms of rollback;
2. Any variable which appears only in SoI_R for all LPs (agents) over $[t_1, t_2]$ (i.e., no agent writes the variable) is not important in terms of rollback either; and
3. A variable which is present in the SoI_R of one LP (agent) and in SoI_W of another may cause a rollback.

A rollback will occur if the variable is in both sets and a late write arrives from one LP after a read was made by another LP (as described in the example above). The third point

above requires that a minimum of two LPs be involved; intuitively as the number of LPs involved increases so does the likelihood of rollback. We can therefore predict the likelihood of rollback using the access patterns of any particular variable.

	Many Writes	Few Writes
Many Reads	High	Medium
Few Reads	Medium	Low

Table 2: How probability of rollback due to a particular variable is affected by reads and writes

1. A variable which is in both SoI_R and SoI_W for many different LPs (agents) – High probability of rollback
2. A variable which is in the SoI_R of a single LP (agent) and in many LPs SoI_W – Medium probability of rollback
3. A variable which is in the SoI_W of a single LP (agent) and in many LPs SoI_R – Medium probability of rollback
4. A variable which only appears in the SoI_R of one LP and in the SoI_W of another – Low probability of rollback

4. AN ADAPTIVE MECHANISM

Table 2 uses the notion of SoI to give an indication of how likely rollback is due to different access patterns of a particular variable. This suggests a simple scheme where the optimism of an LP should reflect the access patterns of the variables in its SoI, the more common variables, the less optimistic. Figure 1 illustrates this idea. There are four agents (LPs) represented by small squares. The circles surrounding each of the agents are an abstraction of their spheres of influence which are defined in table 3 [1]. Circles overlapping indicates that the two agents read and write some common variables. With the assumption that each agent reads or writes a variable within its sphere of influence with equal probability over the time interval $[t_1, t_2]$, a larger overlap indicates that the two agents access more common variables.

Agent	Sphere of Influence
A1	x_1, x_2, x_3, x_4, x_8
A2	x_1, x_5, x_6, x_8
A3	$x_2, x_3, x_4, x_5, x_6, x_7, x_8$
A4	x_7

Table 3: Agents sphere of influence.

Agent A4 reads and writes the smallest number of common variables (smallest intersection) and under the suggested scheme would be given the largest time window and hence would execute with the highest degree of optimism. Agent A3, however, reads and writes the largest number of common variables (it's circle intersects with the other three) and would be given the smallest time window. Hence agent A3 would execute with the least degree of optimism (most conservative). If we assume a balanced load (or at least not a highly imbalanced load) then A4 will execute a faster rate than A3. As a result, the difference in LVT between A3 and A4 will increase with time as shown in Figure 2 (here A3 and A4 have a time window of 5 and 20 respectively) .

[1]For this example we make the valid simplifying assumption that SoI_R=SoI_W

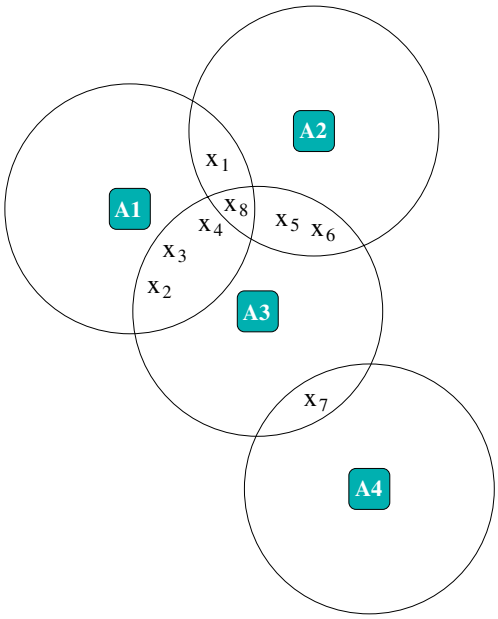

Figure 1: Four agents and the intersection of their spheres of influence

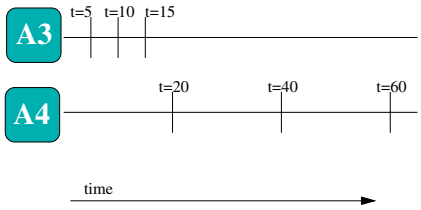

Figure 2: Progression of LVT with time windows

A4 and A3 read and write some common variables (x_7), so the fact that they execute with different degrees of optimism increases the chance of rollback, e.g., A4 would have to rollback if it reads x7 at virtual time t=40 and A3 subsequently (in real time) writes x7 at virtual time t=15. Clearly it is not desirable to have LPs which read and write common variables executing at extremely different rates. If the SoI of two agents A_i and A_j overlap then we can say the probability of A_i causing rollback on A_j is affected by the number of *critical accesses* made between A_i and A_j (CA_{ij}) defined as;

$$CA_{ij} = |SoI_{Ri} \cap SoI_{Wj}| + |SoI_{Wi} \cap SoI_{Rj}| \quad (1)$$

We can then say for any agent A_i (or LP_i, the terms can be used interchangeably here) the likelihood of rollback is affected by all critical accesses made between A_i (CA_i) and all other $n-1$ agents,

$$CA_i = \sum_{j=1}^{n-1} (CA_{ij}) j \neq i \quad (2)$$

The size of the time window for a given agent A_i is therefore

inversely proportional to

1. the number of critical accesses in it's sphere of influence, CA_i; and
2. for each neighbouring agent A_j (i.e., $CA_{ij} \neq 0$) whose LVT differs from A_i by ΔLVT_{ij}, $\Delta LVT_{ij} \times CA_{ij}$

We can now state the form of the equation used to determine window size (optimism) of an agent A_i,

$$WS_i = \frac{a}{k_1 CA_i \times \sum_{j=1}^{n-1} (k_2 CA_{ij} \times k_3 \Delta LVT_{ij})} \quad (3)$$

Where the total number of agents in the simulation is n and a, k_1, k_2, k_3 are appropriate constants.

To enable each LP to calculate its time window we need to have global information regarding the access patterns on shared state variables and the LVTs of the LPs in the simulation. We now propose a simple centralised scheme for collecting the relevant information. First we allocate a counter to each variable in the shared state. This counter indicates how many different LP's spheres of influence the variable lies in and so indicates how difficult the variable is to associate with a particular LP. A central LP is used to collect this information, with all reads and writes passed via this central LP[2]. The LP would simply keep a list of all access made for each variable in the shared state (similar to Table 1) between a time period $[t_1, t_2]$. From this, the centralised LP can determine CA_i and CA_{ij} for all LPs in the simulation. It can also determine the current LVT of each LP (via the time stamp of the most recent access) and hence ΔLVT_{ij} for all LPs i and j.

At time t_2 a GVT computation would occur and the new window size would be calculated for all LPs. This relies on the fact that the access patterns from the time period $[t_1, t_2]$ will produce an appropriate window for the time period $[t_2, t_3]$. If the length of the time intervals are chosen appropriately, the change in the spheres of influence from one time period to the next should be small. It was shown in [Logan and Theodoropoulos, 2001] for some typical agent simulations the change in spheres of influence is limited.

Using a central controller LP in this way limits the message traffic. Without a central controller each LP would need to broadcast all information to the other LPs in the simulation. A protocol which uses global information in this way incurs extra overhead, and we envisage that the development of CLPs will offer a solution to this problem.

5. RESULTS

Investigation of the metric is still at a preliminary stage and our results to date relate to the spheres of influence rather than the performance of the metric itself. These experiments and results serve as feasibility study for later development of the metric. The experiments here are performed in the SIM_TILEWORLD [Lees, 2002] testbed implemented in the agent toolkit SIM_AGENT. Tileworld is a commonly used testbed in agent evaluation and experimentation. The Tileworld environment consists of tiles, holes and obstacles. A Tileworld agent tries to score as many points as possible by filling holes with tiles (the agent receives a point for each tile placed in a hole). The Tileworld environment (see figure 3) is dynamic in that objects (holes, tiles and obstacles) are created at random with a predefined lifespan. The experimenter

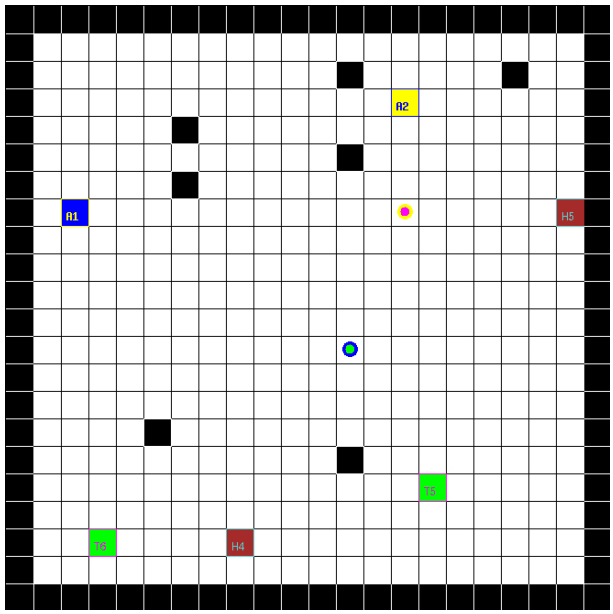

Figure 3: The SIM_TILEWORLD testbed

can control how dynamic the environment is by defining the probability of new objects being created and the lifespan of those objects. For example, an environment with high object creation probability and short lifespan would be very dynamic. Tileworld also allows the experimenter to vary the density of objects in the environment: if the object creation probability is high and the lifespan is high there will be a large number of objects in the environment at any one time.

The results presented below show how access patterns vary when the density of objects in the environment is changed. In particular the experiments look at how access patterns relate to the number of possible rollbacks occurring in a simulation. The initial hypothesis being that as the number of common accesses increases so does the number of possible rollbacks occurring. Possible rollbacks are identified by a particular access pattern. Firstly a ΔLVT value l is set, this defines the largest possible difference between the time stamp of the access and the LVT of the receiving LP[3]. A possible rollback pattern occurs if a variable is written to by one agent and then read by another agent up to l time periods (cycles) later. In these experiments we set l to be 3.

To explain the results we first introduce the notion of a common read and common write. A *common read* occurs when one agent reads to a variable and another agent also accesses the same variable (i.e., read or write). A *common write* occurs when one agent writes to a variable and another agent also accesses the same variable (i.e., read or write). The first graph (figure 4) shows the read patterns of two agents in a 20x20 Tileworld environment during a 20 cycle period. Each agent has a sensor range of 5 squares. The graph shows how

[2]This LP behaves much the same way as the CLPs described in [Logan and Theodoropoulos, 2001]

[3]This value should reflect typical differences in LVT of two LPs

the number of reads made by each agent and the common reads varies with the number of objects in the environment. With an object creation probability of 0.1 both agents made about 300 reads, a high percentage of these were common reads (200 or 66%). At 0.1 object creation probability there are few objects in the environment in this situation the agents tend to aim for the same tiles and holes. As the number of objects in the environment increases the proportion of common reads drops (to around 44%). Figure 5 shows how the number of common reads varies with the size of the environment. With the environment at size 80x80 the number of common reads has almost dropped to zero. This could be due to two things, firstly the agents are further apart and secondly the environment is less densely populated with tiles and holes.

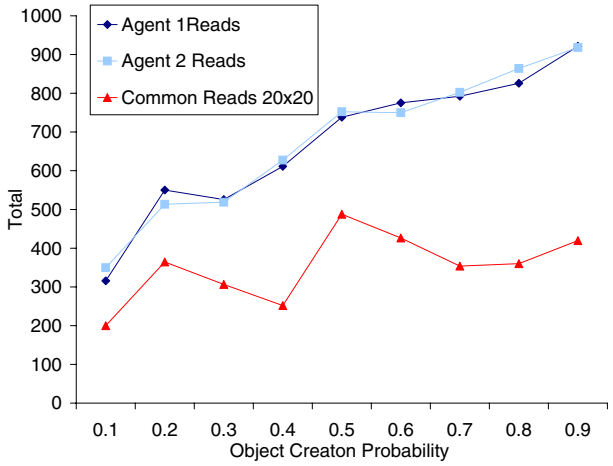

Figure 4: The reads made by two agent in a 20x20 Tileworld environment in a 20 cycle period

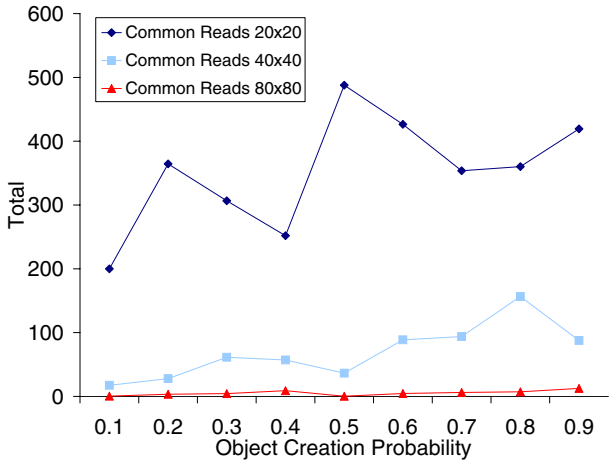

Figure 5: How the number of common reads made between two agents varies with object creation probability and environment size

The graphs in Figures 6-8 show the write patterns of two agents in environments of varying size. The graphs also show how closely related rollback patterns and common writes are.

From this we can say that rollbacks patterns will occur when

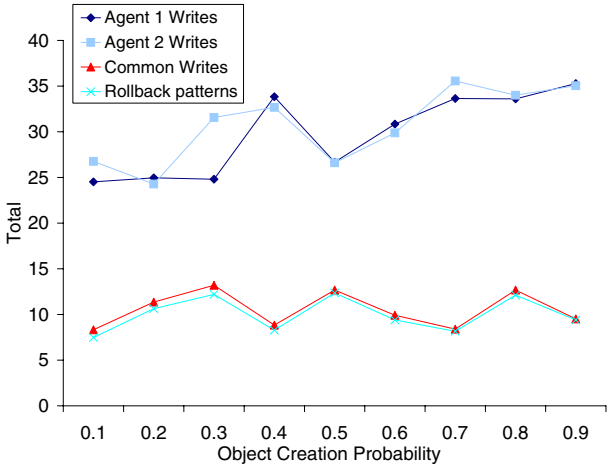

Figure 6: Writes and rollback patterns of two agents in a 20x20 Tileworld environment over a 20 cycle period

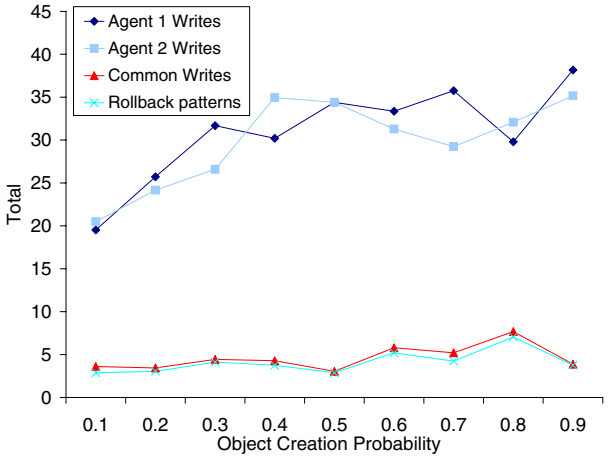

Figure 7: Writes and rollback patterns of two agents in a 40x40 Tileworld environment over a 20 cycle period

the activities of the agents result in a common write. For example, a rollback pattern will occur if agent A_1 is pushing a tile which is within the sensor range agent A_2 up to 3 cycles (moves[4]) later. This conclusion is reinforced upon comparison of the number of rollback patterns in different sized environments. As the environment size increases the number of writes made by each agent drops. The number of common writes drops even further and hence rollback patterns become much less common in larger, less dense environments.

6. DISCUSSION AND FUTURE WORK

In this paper we describe a novel adaptive optimistic synchronisation mechanism for the parallel discrete event simulation of agent-based systems. Our mechanism uses the Sphere of Influence of an event to define an adaptive metric which can be used with a throttling mechanism such as moving time windows. We show how such a metric can be calculated by monitoring the common reads and writes made by the

[4]The agents currently implemented in SIM_TILEWORLD are purely reactive and hence move one square every cycle

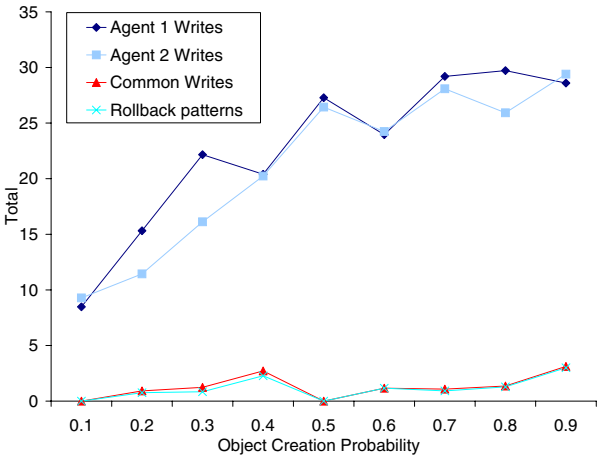

Figure 8: Writes and rollback patterns of two agents in a 80x80 Tileworld environment over a 20 cycle period

agents to the shared simulation state modelling the agent's environment, and present the results of our preliminary investigations into the relationships between agent read and write patterns and rollback frequency.

Our experimental results show, as expected, that the number of common accesses does affect the number of rollback patterns occurring in an agent simulation. Surprisingly, the results have shown that for this particular agent simulation (SIM_TILEWORLD) rollback patterns are very closely related to common writes. This relates to the ratio of writes to reads made by the agents. If, as in this case, the agents perform a large number of reads compared to writes (700/35), then almost all common writes will result in a rollback pattern. Our plans for further work in this area will be toward investigating this relationship. We plan to repeat the experiments with different types of agent simulation with highly different access patterns.

Our long term goal is to implement the adaptive metric in Georgia Time Warp (GTW). Initially a simple C/C++ program will simulate typical access patterns of agent simulation (mirroring the results obtained here). This allows us to test the metric without the need for integrating an agent toolkit with GTW beforehand. If the results are as expected the next step is to integrate an agent toolkit (e.g., SIM_AGENT) with GTW to investigate the performance of the mechanism.

Acknowledgements

We would like to thank Dr Bora Kumova and Tonworio Oguara, both based at the University of Birmingham, for their contribution to this paper. This work is part of the PDES-MAS project[5] and is supported by EPSRC research grant No. GR/R45338/01.

7. REFERENCES

Ferscha, A. (1995). Probabilistic adaptive direct optimism control in time warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pages 120–129.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, pages 558–564.

Lees, M. (2002). A history of the tileworld agent testbed. Technical report, Nottingham University.

Logan, B. and Theodoropoulos, G. (2001). The distributed simulation of multi-agent systems. In *Proceedings of the IEEE*, pages 174–185.

Sokol, L. and Stucky, B. (1990). Mtw: Experimental results for a constrained optimistic scheduling paradigm. In *Proc. SCS Multiconf. Distributed Simulation*, volume 22, pages 169–173.

[5]http://www.cs.bham.ac.uk/ gkt/Research/PDES/pdes-mas.html

Author Biographies

MICHAEL LEES. is a PhD student studying in the School of Computer Science and IT at the University of Nottingham, UK. He received a joint Honours Computer Science and Artificial Intelligence degree from the University of Edinburgh,UK in 2001. His thesis will be centred of the areas of Multi-Agent systems and distributed simulation.

BRIAN LOGAN. is a lecturer in the School of Computer Science and IT at the University of Nottingham, UK. He received a PhD in design theory from the University of Strathclyde, UK in 1986. His research interests include the specification, design and implementation of agent-based systems, including logics and ontologies for agent-based systems and software tools for building agents.

GEORGIOS THEODOROPOULOS. received a Diploma degree in Computer Engineering from the University of Patras, Greece in 1989 and MSc and PhD degrees in Computer Science from the University of Manchester, U.K. in 1991 and 1995 respectively. He is currently a Lecturer in the School of Computer Science, University of Birmingham, U.K. His research interests include parallel and distributed systems, computer and network architectures and modelling and distributed simulation. He is a co-founder of the Midlands e-Science Center of Excellence in Modelling and Analysis of Large Complex Systems.