Analysing Probabilistically Constrained Optimism

Michael Lees and Brian Logan
School of Computer Science & IT
University of Nottingham UK
{mhl,bsl}@cs.nott.ac.uk

Dan Chen, Ton Oguara and
Georgios Theodoropoulos
School of Computer Science
University of Birmingham, UK
{cxd,txo,gkt}@cs.bham.ac.uk

Abstract

In previous work we presented the DTRD algorithm, an optimistic synchronisation algorithm for parallel discrete event simulation of multi-agent systems, and showed that it outperforms Time Warp and time windows on range of test cases. DTRD uses a decision theoretic model of rollback to derive an optimal time to delay read event so as to maximise the rate of LVT progression. The algorithm assumes that the inter-arrival times (both virtual and real) of events are normally distributed. In this paper we present a more detailed evaluation of the DTRD algorithm, and specifically how the performance of the algorithm is affected when the inter-arrival times do not follow the assumed distributions. Our analysis suggests that the performance of the algorithm is relatively insensitive to events whose inter-arrival times are not normally distributed. However as the variance of the input events increases its performance degrades to that of Time Warp. Our approach to evaluation is general, and we outline how the analysis may be applied to other decision theoretic algorithms.

1 Introduction

A Parallel Discrete Event Simulation (PDES) consists of a series of Logical Processes (LPs) each executing in parallel. Events are generated and processed by the LPs and an LP will typically schedule events on the other LPs in the system. *Synchronisation* ensures that events in a parallel simulation are processed in the correct causal order. One of the challenging problems of PDES is the design and evaluation of synchronisation mechanisms. Synchronisation algorithms can be classified into two categories depending on how they enforce the local causality constraint. *Conservative* algorithms strictly adhere to the local causality constraint by ensuring events are processed only when they are guaranteed not to violate causality. *Optimistic* synchronisation algorithms allow violations of causality but provide a mechanism (rollback) to undo and repeat invalid computation.

Both optimistic and conservative approaches have advantages and disadvantages. Hybrid schemes try to combine the benefits of each approaches while limiting their negative effects. The performance of these constrained or bounded optimism approaches depends greatly on the metrics used by the algorithm and its ability to adapt to changes in the simulation. It is often difficult to determine an ideal degree of optimism and also to verify that the algorithm effectively constrains the optimism of the simulation.

In previous work [4, 5] we presented the DTRD algorithm, a throttling mechanism for parallel discrete event simulation of multi-agent systems (MAS). DTRD uses a decision theoretic model of rollback to derive an optimal time to delay an event so as to maximise the rate of local virtual time (LVT) progression. The algorithm assumes that the inter-arrival times (both virtual and real) of events are normally distributed. Using this assumption the algorithm determines the probability of rollback using the history of recent arrival times. In [4] we showed that DTRD outperforms Time Warp and time windows on range of stereotypical multi-agent simulation test cases. In this paper we present a more detailed evaluation of the DTRD algorithm focusing on two key questions: how the algorithm performs as the variance of the input events increases (i.e., as the events become less predictable); and how the algorithm performs when the real and virtual inter-arrival times are not normally distributed.

The remainder of the paper is organised as follows. In section 2 we describe the DTRD algorithm and show how it uses the distribution of inter-arrival times to compute an optimal delay time for an event. In section 3 we present a detailed evaluation of the DTRD algorithm and show how its performance is affected when the variance of the input events increases and when the real inter-arrival times are not normally distributed. In section 4 we discuss some related probabilistic synchronisation algorithms and briefly outline how the same evaluation technique can be applied to these algorithms. Finally, in section 5, we conclude and outline plans for future work.

2 Decision-Theoretic Throttling

In [6] we presented the PDES-MAS framework for the distributed simulation of multi-agent systems. In PDES-MAS LPs are divided into two categories, *Agent Logical Processes* (ALPs) and *Communication Logical Processes* (CLPs). ALPs contain the agent models and the private state of the agents, while the CLPs contain *state variables* which hold the public state of the simulation. During the simulation ALPs interact with the public state by sending read and write events to the CLPs.

A defining characteristic of agents is their autonomy. The actions performed by an agent are not solely a function of events in its environment: in the absence of input events, an agent can still produce output events in response to autonomous processes within the agent. As a result, agent simulations have zero lookahead [10]. The PDES-MAS framework therefore utilises an optimistic synchronisation strategy. However, unconstrained optimism can result in excessive rollback [9]. One approach to reducing rollback is throttling, or bounded optimism, in which LPs are prohibited from processing events which are likely to be rolled back.

In [4, 5] we presented the DTRD algorithm, an adaptive optimistic synchronisation algorithm for PDES-MAS. DTRD attempts to maximise the rate of LVT progression (defined as virtual time / elapsed time) for each ALP, taking into account the likely behaviour of the other ALPs. Intuitively, we want to advance LVT safely (i.e., without rollback) to the time of the next read event in the smallest possible amount of elapsed time. In the context of PDES-MAS, rollback occurs when a straggler write is received by a CLP which invalidates a read previously made by an agent. In conventional optimistic PDES, rollback has generally been considered in terms of late or *straggler* messages. This seems intuitive given the overall goal, which is to make the simulation execute as fast as possible. However we can also say that rollbacks result from processing an event prematurely. We say an event e with time stamp t_e is *premature* if another event e' with timestamp $t_{e'} < t_e$ will arrive after e in real time.

DTRD attempts to maximise the rate of LVT progression by delaying reads which are likely to be premature. The algorithm calculates the optimal time to delay a premature read of a state variable based on the history of writes to the variable. DTRD uses a decision theoretic approach to determine the ideal trade off between allowing optimistic execution and preventing rollback. The algorithm is presented with $n + 1$ different decisions, whether to delay the read for $0, j, 2j, \ldots, nj$, where j is some amount of real time. Each decision has an overall expected utility (or cost) which is measured units of real time. The optimal delay time is then the decision with lowest expected cost, giving an ideal trade-off between delays and optimistic execution.

To calculate the overall cost of each delay action, A_j, the algorithm needs to calculate:

$$\begin{aligned}&EC(A_j|E) = \\ &\quad P(NoStraggler|E) \times C(NoStraggler) + \\ &\quad P(Straggler|E) \times C(Straggler).\end{aligned}$$

$P(Straggler|E)$ is the probability that a straggler write will arrive after a delay of j, where E is the evidence (history of writes to the state variable). $P(NoStraggler|E)$ is the probability that no straggler write will arrive after a delay of j and can be calculated as $1 - P(Straggler|E)$. $C(NoStraggler)$ is simply the delay time for action A_j, i.e., j. $C(Straggler)$ is the sum of the delay time for action A_j plus the rollback cost c, i.e., $j + c$. The optimum action is the one with the lowest expected cost. Figure 1 shows the minimum expected cost for a single read for varying delay times. As can be seen, as the delay time increases, the expected cost of a straggler write decreases while the expected cost of no straggler increases. The total expected cost is the sum of these two curves, and the optimum delay is that which minimises the total expected cost (in this example approximately 250 msecs).

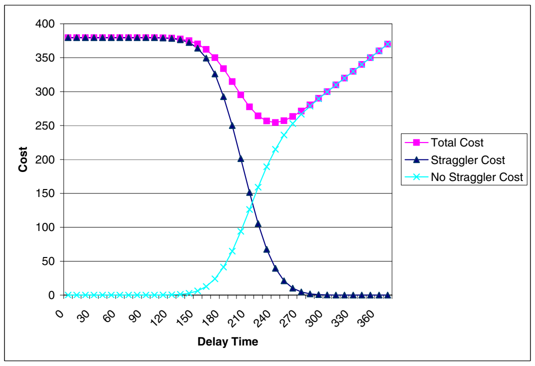

Figure 1. Minimum expected cost for a single read (variance 30)

Given the above the algorithm can be broken down into two calculations: firstly determining the cost of rollback, and secondly determining the probability of rollback. The cost of rollback is calculated by estimating the real time cost of undoing prematurely committed events and replaying the invalid computation. With 100% processor utilisation, CPU time spent in rollback is equivalent to elapsed time spent delaying: with less favourable processor utilisation (e.g., if we only have access to 50% of the processor time) the cost of rolling back will be proportionately higher in elapsed time. We therefore estimate the cost of rollback directly in terms of elapsed time, rather than CPU time. To calculate the probability of rollback the algorithm assumes underlying distributions for the both the real and virtual inter-arrival times of write events. Using the timestamps of previous write events it is possible to determine the mean and standard deviation of these distributions, and so calculate the probability of rollback for a read r, with real and virtual arrival time a_r and t_r. To do this we calculate the probability that the real and virtual timestamps of the next write, a_w and t_w, are such as to cause a rollback, i.e., $a_r + j < a_w$ and $t_r > t_w$ (see [4] for details).

3 Evaluating the DTRD algorithm

In previous work [4] we compared the performance of the DTRD algorithm with that of Time Warp and time windows on both stereotypical cases designed to be representative of the kinds of access patterns found in multi-agent simulations and on traces from a real agent simulation. For the first set of experiments, we characterised the agents' interaction with the shared state of the simulation in terms their degree of coupling and their skew. *Coupling* refers to how the ALPs interact with the shared state variables and the resulting potential for causality violations (rollback). By *skew* we mean the difference in the 'natural' rate of LVT progression between the ALPs, in the absence of rollback or any throttling mechanism. A given set of ALPs may vary between high and low degrees of coupling throughout the execution of the simulation. Similarly, depending on the execution environment (system loads etc) the degree of skew may vary throughout the simulation.

We tested the DTRD algorithm on a range of synthetic event traces derived from both fully-coupled and uncoupled systems with varying degrees of skew. The algorithm performed well on these test cases, adapting to differing degrees of both coupling and skew. For example, DTRD has similar performance to Time Warp and outperforms time windows in the uncoupled case and outperforms both algorithms in the fully-coupled case (87% fewer replayed cycles and 30% less CPU time). To evaluate the algorithm's performance in intermediate coupling cases, we also compared the performance of the DTRD algorithm with that of Time Warp and time windows on traces taken from the Boids [8] agent simulation benchmark. We showed that the performance of DTRD was at least as good as Time Warp and time windows in terms of computation time in all the cases investigated, and often significantly better.

While these results are encouraging, the performance of the DTRD algorithm depends on the algorithm's ability to exploit predictability in the arrival time of the next write event. In our previous work, we focused on the structural characteristics of the agent simulation (i.e., coupling and skew), and considered only a limited range of event distributions. In particular, in the experiments referred to above, both the real and virtual inter-arrival times were assumed to be normally distributed. The virtual inter-arrival times were assumed to have a mean of 15 and standard deviation of 4, and the mean real inter-arrival time was 50 msecs for the slower agent and $50 \times \mathrm{skew}$ msecs for the faster agent, with a standard deviation for both agents of 0.05 times the mean.

In this paper, we present a more detailed analysis of the performance of DTRD algorithm. We focus on two key questions:

- how the algorithm performs as the variance of the input events increases (i.e., as the events become less predictable); and
- how the algorithm performs when the real and virtual inter-arrival times are not normally distributed.

3.1 Method

To analyse the performance of the algorithm we developed a meta-simulator using Microsoft Excel. The meta-simulator abstracts the ALPs to a list of read and write events. The real and virtual inter-arrival times between these events are generated from various probability distributions. The meta-simulator proceeds by calculating the next event to be processed. This is done by determining which ALP's next event has the lowest real arrival time. The virtual time of the next event is then calculated, and, if the event causes a rollback, the other ALP's event trace is reset to the time of the rollback. The total time for the simulation is then the total time required to process all events and any rollbacks.

The DTRD algorithm is also implemented within the Excel meta-simulator. When DTRD is enabled, the optimal delay time for each read event is computed and the read event requeued for processing at *now* + the chosen delay time. The implementation of the DTRD algorithm uses the same method for calculating the cost and probabilities as used in [4]. For comparison purposes, we also implemented a simple version of Time Warp within the meta-simulator, which simply processes each read event immediately, i.e., no delays are applied. The meta-simulator progresses until all events have been executed, measuring the number of replayed events, rollbacks, elapsed time and total delay time applied.

We used a single, half-coupled test case [4] for all experiments, in which two agents, each running on a separate ALP, access a single shared state variable. Each agent executes a *sense–think–act* cycle, with read and write events corresponding to the sensing and acting phases of the agent's cycle. At each cycle, agent 1 reads the variable, and agent 2 both reads and writes the variable. In all cases, agent 2 has a lower rate of LVT progression than agent 1. Agent 2 therefore constrains the optimism of agent 1 and determines the rate of global virtual time (GVT) progression of the simulation as a whole. Although the scenario is simplified, this pattern of variable accesses is representative of the kinds of interactions found in highly coupled agent simulations, where agents interact via the shared state. For example, in flocking or predator and prey simulations in which one agent can sense another, but the second agent cannot sense the first [8].

We determined the performance of the algorithms both for normally distributed event sequences with different variances and for event sequences which are not normally distributed. We used the total number of cycles repeated by all agents as a result of rollback as our primary performance measure.[1] This value indicates the frequency and depth of rollbacks within the system. For each algorithm, we report the number of replayed cycles averaged over five input traces for each set of test parameters.

3.2 Increased Variance

For the first set of test cases the input events were normally distributed with a fixed mean and increasing standard deviation. The mean real inter-arrival time was taken to be 100 msecs for the faster agent and 200 msecs for the slower agent, and the standard deviation was taken to be 5, 10, 15, 20 and 25% of the mean for each agent. The mean virtual inter-arrival time was 15 for both agents, and the standard deviation was varied from 5 to 25% of the mean. This scenario is intended to be representative of simulations in which the agents have approximately the same virtual cycle time, but different real cycle times (either because of differences in CPU availability or because the amount of CPU required by the 'think' part of the agent cycle is different for each agent).[2]

This set of test cases is intended to evaluate the performance of the DTRD algorithm when the underlying assumption of normality does hold but increased variance makes it harder to predict the real and virtual timestamps of the next write event. Figure 2 shows the number of replayed events for the DTRD algorithm as the variance of the real and virtual timestamps increases. As can be seen the number of replayed cycles increases from 20.6 when the real and virtual variance are 5% of the mean to 55.2 when the real and virtual variance are 25% of the mean. Increases in the variance of real inter-arrival times has a more marked effect on the performance.

[1]Elapsed time is not informative in this case, as the last event generated by the slower agent determines the minimum execution time.

[2]We focus on variation in real cycle times as large differences in virtual cycle times are less common in simulations of situated agents.

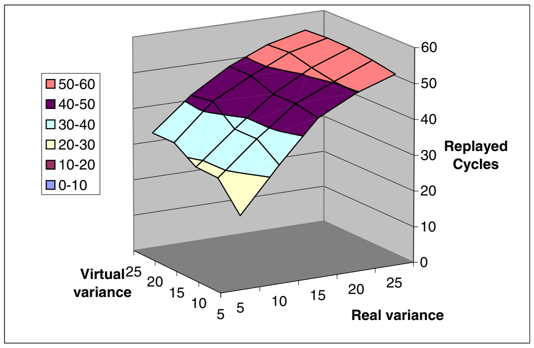

Figure 2. Performance of DTRD with increasing variance

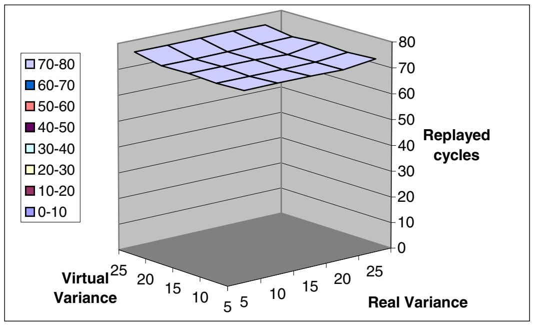

Figure 3. Performance of Time Warp with increasing variance

For comparison, Figure 3, shows the number of replayed events for Time Warp as the variance of real and virtual timestamps increases. As can be seen, the unconstrained optimism of Time Warp is relatively insensitive to increases in the variance of the real and virtual inter-arrival times. For example, the number of replayed cycles increases from 73.2 when the real and virtual variance are 5% of the mean to 75.6 when the real and virtual variance are 25% of the mean.

These results indicate that DTRD is able to exploit predictability in the event sequences to reduce the number of rollbacks and replayed cycles. This is consistent with results previously reported in [4, 5]. However, as the variance increases, i.e., as it become more difficult to predict the real and virtual timestamps of next write event, the performance of DTRD degrades until it approximates that of Time Warp. Figure 4 shows how the average total delay time for all events decreases as the variance increases. When the real and virtual variance is 5% the total delay time is 7041.4 msecs, decreasing to 2236.7 msecs as the real and virtual variance increases to 25%.

As the variance increases, the reduction in expected cost associated with non-zero delays reduces, and eventually dis-

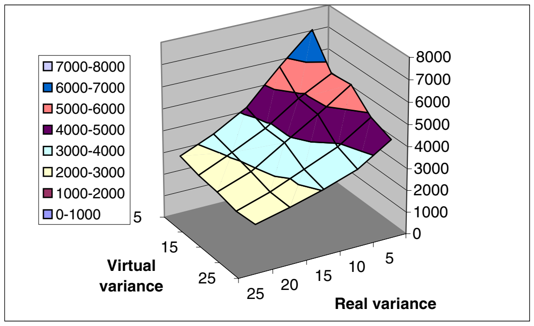

Figure 4. Total delay time for increasing variance

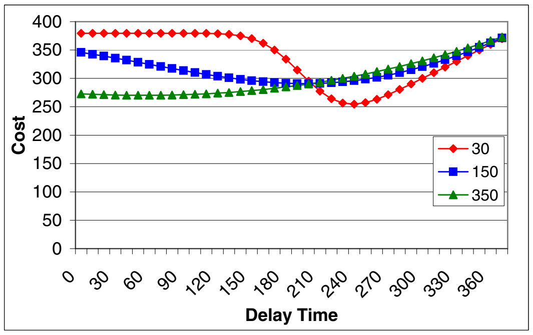

Figure 5. Minimum expected cost for a single read for increasing variance

appears entirely. As an example, Figure 5 shows the expected cost computed by the DTRD algorithm for a single read by agent 1 (i.e., for a mean real inter-arrival time of write events of 200 msecs) where the standard deviation of the real inter-arrival time of writes is 30, 150 and 350 msecs. In this example the mean real inter-arrival time of writes was calculated as 202.27 msecs and the mean virtual inter-arrival time is 14.95 with standard deviation 1. The timestamp of the last write is 175 and the timestamp of the read is 205. As can be seen, when the variance is low, delaying until after the expected time of the next write event gives the minimum expected cost. However, as the variance increases, the reduction in expected cost associated with non zero delays becomes smaller until with a standard deviation of 350, there is no expected benefit to be obtained from delaying.

3.3 Non-Normality

A key assumption of the algorithm is that both real and virtual inter-arrival times are normally distributed. In this section we report experiments designed to evaluate the performance of the algorithm when this assumption does not hold.

We generated event sequences in which the real and virtual inter-arrival times were drawn from a triangular and a uniform distribution. The distributions were chosen so as to have the same mean and standard deviation as the normal distributions used for the previous experiments. For the uniform distribution the size of the outcome space was taken to be $\mu \pm (\sqrt{3}\mu \times \sigma_n)$ and for the triangular distribution the end points were $\mu \pm (\sqrt{6}\mu \times \sigma_n)$, where σ_n is the standard deviation of the corresponding normal distribution. For any given read event, the algorithm should therefore choose the same delay time as for normally distributed data, allowing the performance of the algorithm to be directly compared with the normally distributed case.

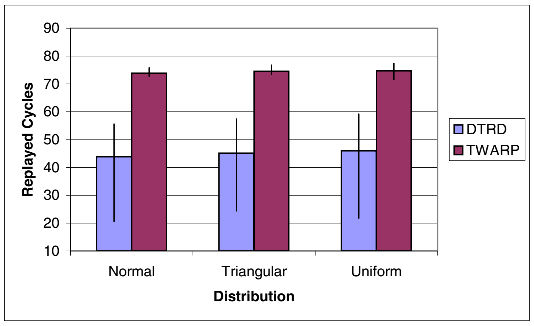

Figure 6. Performance of Time Warp and DTRD for non-normally distributed events

Figure 6 shows the average, minimum and maximum number of replayed cycles for the normal, triangular and uniform distributions for both DTRD and Time Warp. As might be expected, varying the distribution of the real and virtual times of events has relatively little impact on the performance of Time Warp: for all distributions, the average number of replayed cycles is around 75, though with uniformly distributed data, the minimum values are slightly lower and the maximum values slightly higher. More surprisingly, the data also suggest that varying the distribution has relatively little impact on the performance of DTRD. The average number of replayed events increases from 43.9 in the normal case to 46 when the events are uniformly distributed. Similarly, the minimum and maximum values for the normally distributed case are 20.6 and 55.6 and for the uniform case are 21.8 and 59.2. The variation in performance is also similar to that for the normally distributed case, i.e., the number of replayed cycles increases with the variance of the input data.

We also investigated the performance of the DTRD algorithm when the real and virtual inter-arrival times are exponentially distributed. Figure 7 shows the average, minimum and maximum number of replayed cycles for DTRD and Time Warp for events chosen from an exponential dis-

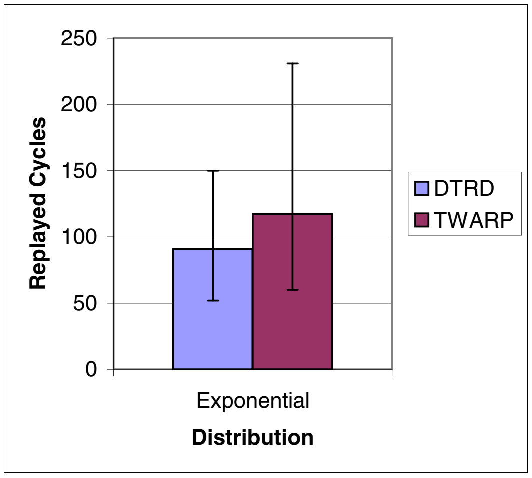

Figure 7. Performance of Time Warp and DTRD for exponentially distributed events

tribution with the same mean real and virtual inter-arrival times as in previous cases. As can be seen, the performance of both algorithms is significantly worse than for the normal, triangular or uniform distributions. For example, with Time Warp, the average number of replayed events increases from 73.9 for normally distributed to data, to 117.4 in the exponential case. With DTRD the average number of replayed events increases from 43.9 in the normal case to 90.8 in the exponential case. However, even with exponentially distributed data, DTRD still outperforms Time Warp. We believe the decrease in performance of DTRD can be attributed to the increased variance of the exponentially distributed data, rather than to the characteristics of the exponential distribution per se.

In summary, the performance of the algorithm does not appear to degrade significantly when the data are not normally distributed. Indeed the results presented above tend to support the hypothesis that the key factor is the variance of the data rather than the assumption of normality. For all distributions, as the variance increases, DTRD delays fewer events, and, in the limit, its performance degrades gracefully to that of Time Warp. Importantly, in none of the cases investigated did DTRD perform worse on average than Time Warp. This is not to suggest that the algorithm is optimal for all cases; however it does suggest that for a range of input distributions it can successfully exploit predictability in the data to reduce the number of replayed cycles. By reducing the number of cases where the last event generated by agent 2 triggers a rollback on agent 1, the DTRD algorithm also slightly reduces the overall elapsed time required to process the same set of events. For the normally distributed inter-arrival times the average elapsed time over all values of standard deviation is 20234 msecs for Time Warp and 20153 msecs for DTRD.

4 Related Work

There are a number of probabilistic approaches to constraining optimistic execution in the literature [1, 2]. In this section we look at two other optimistic algorithms which apply similar approaches to the DTRD algorithm. We indicate the assumptions made by the algorithms when computing the probability of rollback, and briefly outline how a similar analysis could be applied.

In [3] a decision theoretic algorithm similar to DTRD is used to calculate the optimal amount of CPU time an LP should delay before processing an event. The algorithm calculates the probability of rollback for a particular event and uses previously observed rollback costs to calculate the delay time. To calculate the probability of rollback, the real-virtual time plane is decomposed into spatial regions $R_{ij} = [s_{i-1}, s_i) \times [t_{j-1}, t_j)$, and the number of events and rollbacks within each region are counted. The frequency of events and rollbacks is not uniform over all real-virtual time regions: assuming distributions of G_{vt} and G_{rt} along the virtual and real axes respectively, the interval boundaries are calculated as

$$s_i = G_{rt}^{-1}\left(\frac{i}{\mu}\right), t_j = G^{-1}vt\left(\frac{i}{v}\right).$$

When considering whether to delay an event, the algorithm determines in which region the event's inter-arrival times lie. The probability of rollback for that event is then the number of rollbacks in the region divided by the total number of events, i.e., the ratio of rollbacks per event.

While the approach presented in [3] does not model the distributions of event times using an explicit distribution, the effectiveness of the algorithm still depends on the distributions used to generate the real and virtual time regions. One may infer that the performance of the algorithm would degrade if the inter-arrival times of events followed a different distribution to the one used to decompose the real and virtual time plane. In [3] the authors indicate that with a uniform decomposition of the real-virtual time plane the algorithms performance is worse. It is not clear how the variance of the distributions would affect this approach. Without the assumption of an underlying distribution the probability of rollback will not necessarily decrease with increased variance. Therefore with high variance the algorithm may still apply delays. However one would expect that with increased variance the delays would be less effective at preventing rollback.

The Minimum Average Cost (MAC) algorithm [7] is another probabilistic synchronisation algorithm which uses a decision theoretic approach to determine delay times. The algorithm is implemented within the ParaSol system, where each LP has an independent input channel for message from every other LP in the system. When the algorithm finds an empty input channel on an LP k, the algorithm determines the amount of real time, w, for which processing on k should be suspended. w is calculated so as to give the optimal trade off between preventing cost of rollback and the cost of delaying.

The MAC algorithm records the real arrival times and timestamps of messages for each channel of an LP. The inter-arrival times are calculated and assumed to be stationary sequences. Given that an input channel is empty at time t, the algorithm calculates the probability that a straggler event will arrive after time $t + w$ as:

$$\begin{aligned} P\{S_j(t+w) = 1\} = \\ P\{T_{j,M} < y_{i,N} - y_{j,M-1} \times \\ R_{j,M} > t + w - x_{j,M-1}\}/ \\ P\{R_{j,M} > t - x_{j,M-1}\} \end{aligned}$$

where $T_{j,M}$ and $R_{j,M}$ are the stationary sequences of virtual and real inter-arrival times, $x_{j,M-1}$ and $y_{j,M-1}$ are the last real and virtual times respectively, and $y_{i,N}$ is the timestamp of the last event to be processed by the LP. The above example only deals with the two source case (i.e., two LPs)—in general the probability of a straggler is calculated as the probability of receiving a straggler on any input channel.

The MAC algorithm makes stronger assumptions about the distribution of inter-arrival times in that it assumes stationarity, i.e., zero variance in the inter-arrival times. With this assumption, for a given mean the probability of receiving a rollback is constant. Therefore if the inter-arrival times were normally distributed, or exponentially distributed with the same mean, the calculated probabilities would be the same. Increasing variance will therefore have no effect on the probability of rollback computed by the algorithm. If there is little variance in the inter-arrival times, the algorithm should see arrival times close to those it predicts. However, with larger variance, the assumption of stationarity would result in many arrival times long before or after the expected arrival time. Therefore, as the MAC algorithm applies the same delays for event sequences with greater variance, we might expect a degradation in the performance of the algorithm in these cases.

5 Discussion and Further Work

In this paper we have presented a detailed analysis of the DTRD algorithm, and showed how its performance changes when different underlying distributions are used to generate the inter-arrival times of events. We first investigated the behaviour of the algorithm as the variance of the distribution of inter-arrival times increases. As the variance increases it becomes harder to predict the real and virtual arrival times of a straggler write, and the algorithm executes more optimistically. In the extreme case (standard deviation 25% of mean) the algorithm's performance is more or less equivalent to that of unconstrained Time Warp. We also investigated the performance of the algorithm when the real and virtual timestamps of the input events are not normally distributed. We showed that if the inter-arrival times follow a different, but comparable[3] distribution the performance of the algorithm is more or less equivalent to the normal case. For exponentially distributed inter-arrival times, the DTRD algorithm still outperformed unconstrained Time Warp, but both algorithms exhibited a marked reduction in performance.

The behaviour of the algorithm can be explained by considering the probability distribution function (pdf) of the normal distribution. As the standard deviation increases, the probability of a variate assuming a value in a given range decreases, resulting in the algorithm estimating a lower probability of rollback. The algorithm therefore chooses smaller delay times, as the expected cost of a straggler event is lower while the cost of the next event not being a straggler remains the same.

In our analysis we have assumed a zero overhead cost. In reality the DTRD will incur a real time overhead for calculating the inter-arrival time distributions and for determining an optimal delay time. In previous work [4, 5] we have shown that, at least for some agent simulations, the performance gain outweighs the overhead associated with the algorithm. However, in general, this may not always be the case, particularly for simulations with a high degree of variance in the inter-arrival times. In such cases it may be possible to reduce the overhead of the DTRD algorithm by assuming a delay time of zero, for example, in situations where the standard deviation of inter-arrival times of writes to a variable exceeds a threshold and the cost of rollback is not too high.[4] The algorithm would still need to calculate the mean and standard deviation of the real and virtual inter-arrival times for each shared state variable, however such an optimisation would remove the overhead of considering non-zero delay times in those cases where a zero delay time would ultimately be selected while still being able to exploit predictability in the case of other variables.

In future work we plan to extended our analysis to the algorithms described in section 4, to allow us to compare their performance to that of DTRD, and to investigate how these algorithms are affected when their underlying assumptions do not hold. Another area of interest is investigating whether the performance of DTRD could be im-

[3]In the sense that the mean and standard deviation can take the same values as a normal distribution

[4]If the cost of rollback is very high, then even with a low probability of rollback the expected cost of a straggler would still be significant.

proved by adapting the assumed distribution to the observed data as proposed in [2]. We also plan to investigate the performance of the DTRD algorithm for multi-modal distributions. At present a multi-modal distribution will be observed as a unimodal distribution with high variance, however in some circumstances it may be possible to model the different modes of the distribution independently.

Acknowledgments

This work is part of the PDES-MAS project[5] and was partially supported by EPSRC research grants Nos. GR/R45338/01, EP/C549406/1 and BBSRC grant No. BB/D006619/1.

References

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

[2] A. Ferscha and G. Chiola. Self-adaptive logical processes: the probabilistic distributed simulation protocol. In *Proceedings of 27th Annual Simulation Symposium*. IEEE Computer Society Press, 1994.

[3] A. Ferscha and J. Luthi. Estimating rollback overhead for optimism control in time warp. In *Proceedings of 28th Annual Simulation Symposium*, 1995.

[4] M. Lees, B. Logan, , C. Dan, T. Oguara, and G. Theodoropoulos. Analysing the performance of optimistic synchronisation algorithms in simulations of multi-agent systems. In S. J. Turner and J. Lüthi, editors, *Proceedings of the Twentieth ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2006)*, pages 37–44, Singapore, May 2006. IEEE-TCSIM, ACM SIGSIM, SCS, IEEE Press.

[5] M. Lees, B. Logan, C. Dan, T. Oguara, and G. Theodoropoulos. Decision-theoretic throttling for optimistic simulations of multi-agent systems. In A. Boukerche, S. J. Turner, D. Roberts, and G. Theodoropoulos, editors, *Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2005)*, pages 171–178, Montreal, Quebec, Canada, October 2005. IEEE Press.

[6] B. Logan and G. Theodoropoulos. The distributed simulation of multi-agent systems. *Proceedings of the IEEE*, 89(2):174–186, February 2001.

[7] E. Mascarenhas, F. Knop, R. Pasquini, and V. Rego. Minimum cost adaptive synchronization: experiments with the ParaSol system. *Modeling and Computer Simulation*, 8(4):401–430, 1998.

[8] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34. ACM Press, 1987.

[5] `http://www.cs.bham.ac.uk/research/pdesmas`

[9] L. M. Sokol, D. P. Briscoe, and A. P. Wieland. MTW: A strategy for scheduling discrete simulation events for concurrent simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, SCS Simulation Series, pages 34–42. Society for Computer Simulation, July 1988.

[10] A. M. Uhrmacher and K. Gugler. Distributed, parallel simulation of multiple, deliberative agents. In *Proceedings of the Fourteenth Workshop on Parallel and Distributed Simulation (PADS'2000)*, pages 101–110, Washington DC, USA, May 2000. IEEE, IEEE Computer Society.