Large-scale Simulations with FLAME

Simon Coakley and Paul Richmond and Marian Gheorghe and Shawn Chin and David Worth and Mike Holcombe and Chris Greenough

Abstract This chapter presents the latest stage of the FLAME development - the high-performance environment FLAME-II and the parallel architecture designed for Graphics Processing Units, FLAMEGPU. The architecture and the performances of these two agent-based software environments are presented, together with illustrative large-scale simulations for systems from biology, economy, psychology and crowd behaviour applications.

1 Introduction

Agent-based systems are widely used in modelling, analysis and simulation of complex and real-life systems. Many agent-based software environments, like MASON [1], Repast [2], Swarm [3], NetLogo [4], FLAME [5] (just to mention a few of them), have been developed in the last decades. The area is fast growing and there are various survey papers on specific topics regarding agent-based systems and their use - agent-based modelling practices, [6], agent-based modelling approaches and associated tools [7] - or on the use of agent-based systems in electricity market modelling [8], manufacturing control [9], hospital environments [10] etc.

FLAME is one of these many agent-based environments which is built based on an underlying formal model, called the X-machine. One of the primary aims of the FLAME architecture is to run on multiple hardware and software platforms and to simulate multi-scale modelling approaches [5].

Over the years, FLAME [5] has evolved based on the requirements of different projects starting from a position-aware framework used for biological agent modelling - keratinocyte colony formation [11]; role of TGF-$\beta 1$ in tissue regeneration

Simon Coakley, Paul Richmond, Marian Gheorghe, Mike Holcombe
University of Sheffield, Sheffield, UK, e-mail: p.richmond@sheffield.ac.uk

Shawn Chin, David Worth, Chris Greenough
Software Engineering Group, STFC Rutherford Apple Labs, Didcot, UK

[12]; lineage dynamics of epidermal stem cells [13]; processes used by mammalian sperm to find the egg [14] - to a position-agnostic framework driven by a static scheduler and message board library catering to economic models - effects of skill upgrading in the presence of spatial labour market frictions [15]; energy shocks and macroeconomic stabilisation policies [16]; and modelling of the European economy [17].

Some of the functionality provided by FLAME was introduced as ad-hoc features for particular models, while many of the architectural design and data structures used within the framework are a direct result of iterative improvements to meet project-specific goals. For the models it was developed for, FLAME was fit for purpose. However, as a generic framework for agent-based modelling there is still much room for improvement.

FLAMEGPU [18] was an early offshoot of FLAME that looked to execute agent-based models on Graphics Processing Unit (GPU) architectures. Although they share a common heritage, models are not fully cross-compatible.

The latest version of the framework, called FLAME-II, expands on FLAME and FLAMEGPU, and creates a backend component which is meant to run on various hardware architectures, including high-performance machines.

In this paper, we briefly describe the design and limitations of the initial FLAME framework, followed by a discussion on the design of the new FLAME-II. The performance of FLAME-II and FLAMEGPU are assessed by using a simple benchmark problem. Finally, the usage of the agent-based approach in solving complex problems is illustrated on a set of case studies from various areas.

2 Preliminary Concepts and the Modelling Paradigm

Complex biological systems require multi-scale modelling where certain aspects of the system are represented at a higher level, whereas others are described at a lower level of detail [19]. The modelling approach, in this case an agent-based model, should be able to provide such multi-scale facilities. The backbone of the agent-based model we present here is represented by a formal model, called an *X-machine*. This state-based model has been initially considered as a generic computational model [20] and later turned into an abstract representation framework of intracellular biochemical interactions [21]. The model was subsequently refined into the so-called *stream X-machine* model [22]. The X-machine model has the ability to capture various levels of details, as we will see below.

The concept of stream X-machine used in this work is from [23].

Definition 1. A *Stream X-Machine* (SXM for short) is an 8-tuple

$$\mathscr{X} = (\Sigma, \Gamma, Q, M, \Phi, F, q_0, m_0)$$

where:

- Σ and Γ are the *input* and *output* alphabets, respectively.

- Q is the finite set of *states*.
- M is the (possibly) infinite set called *memory*.
- Φ is a set of *partial functions* φ; each such function maps an input and a memory value to an output and a possibly different memory value, $\varphi : \Sigma \times M \to \Gamma \times M$.
- F is the *next state partial function*, $F : Q \times \Phi \to Q$, which given a state and a function from the type Φ determines the next state. F is often described as a *state transition diagram*.
- q_0 and m_0 are the *initial state* and *initial memory*, respectively.

The model consists of two key components: (i) an underlying state machine structure, with states, Q, and transitions, F, controlling the behaviour of the model, and (ii) a set of partial functions Φ describing various actions of the model, triggered by inputs, from the set Σ, and acting on memory values, from the set M, and generating in turn outputs, Γ, and updating the memory.

In order to model more complex systems, the stream X-machine model has been extended to *communicating stream X-machine systems*, a formalism consisting of a set of stream X-machines linked through various communication mechanisms. Some variants have been studied [24, 25, 26], exhibiting different communication and synchronisation strategies. The model proposed in [25] consisting of a communication matrix that is utilised for exchanging messages between the SXM components has been used for modelling agents' behaviour in the approach presented below. The formal definition of a communicating stream X-machine system is provided below [25].

Definition 2. A *Communicating Stream X-Machine System* (*CSXMS* for short) with n components is a tuple $S_n = ((Z_i)_{1 \leq i \leq n}, E)$, where:

- $Z_i = (\Sigma_i, \Gamma_i, Q_i, M_i, \Phi_i, F_i, I_i, T_i, m_{i,0})$ is the SXM with number $i, 1 \leq i \leq n$.
- $E = (e_{ij})_{1 \leq i,j \leq n}$ is a matrix of order $n \times n$ with $e_{ij} \in \{0, 1\}$ for $1 \leq i, j \leq n$, $i \neq j$ and $e_{ii} = 0$ for $1 \leq i \leq n$.

A CSXMS works as follows:

- Each individual CSXMS is a SXM plus an implicit input queue (i.e., of FIFO (first-in and first-out) structure) of infinite length; the CSXMS only consumes the inputs from the queue.
- An input symbol σ received from the external environment (of FIFO structure) will go to the input queue of a CSXMS, say Z_j, provided that it is contained in the input alphabet of Z_j. If more than one such Z_j exist, then σ will enter the input queue of one of these in a non-deterministic fashion.
- Each pair of CSXMS, say Z_i and Z_j, have two FIFO channels for communication; each channel is designed for one direction of communication. The communication channel from Z_i to Z_j is enabled if $e_{ij} = 1$ and disabled otherwise.
- An output symbol γ produced by a CSXMS, say Z_i, will pass to the input queue of another CSXMS, say Z_j, providing that the communication channel from Z_i to Z_j is enabled, i.e. $e_{ij} = 1$, and it is included in the input alphabet of Z_j, i.e. $\gamma \in \Sigma_j$. If these conditions are met by more than one such Z_j, then γ will enter the

input queue of one of these in a non-deterministic fashion. If no such Z_j exists, then γ will go to the output environment (of FIFO structure).
- A CSXMS will receive from the external environment a sequence of inputs $s \in \Sigma^*$ and will send to the output environment a sequence of outputs $g \in \Gamma^*$, where $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_n$, $\Gamma = (\Gamma_1 \setminus In_1) \cup \ldots \cup (\Gamma_n \setminus In_n)$, with $In_i = \cup_{k \in K_i} \Sigma_k$, and $K_i = \{k \mid 1 \leq k \leq n, e_{ik} = 1\}$, for $1 \leq i \leq n$.

The CSXMS formalism and the SXM model provide, through abstractions like X-machine networks and functions, adequate mechanisms for expressing multi-scale modelling.

Before being used in connection with various agent-based approaches the X-machine model and CSXMS have been used as models of intracellular biochemical interactions [21], bee collective foraging [27], monomorium pharaonis ant colonies behaviour [28], rice blast fungus [29].

The X-machine model is at the heart of the FLAME agent-based environment [5] – a thorough presentation can be found in [30]. More recently, the NetLogo environment [4] has been extended with a library of state machine and X-machine primitive functions [31, 32, 33].

3 FLAME and FLAMEGPU Architectures

In this section we present the key features of the agent-based environments, FLAME and FLAMEGPU, developed so far. We describe the usage of the X-machine model as a fundamental part of them, the main differences between these two software environments and the latest developments. These two software environments make use of the concept of CSXMS [25] – see Definition 2.

3.1 Overview of the Design of FLAME

FLAME allows to define agents using the concept of CSXMS, but it also provides mechanisms to incorporate models described in CellML, SBML, or as sets of differential equations. Our aim was to devise a generic modelling technique which enables modellers to define applications as populations of agents interacting and dynamically changing their status or by easily importing and plugging into an agent-based representation models expressed in other specification languages. To enable wide applicability, we have adopted a modular and flexible approach to link our agent-based modelling environment FLAME with existing tools such as COPASI [34] and JSim [35].

In FLAME, modellers define agents utilising an acyclic state machine that characterises the behaviour of the agent per iteration. Each state transition function has access to the internal memory of the agent, as well as input and output streams of information, as these are presented in Definition 1. In FLAME the input/output

FIFO channels and the communication matrix E - see Definition 2 - take the form of message boards [36]. Since message boards are the only means in which the agent communicates with the environment and other agents, this makes the agent model inherently parallel. Each agent can be executed independently as long as the input message board contains the expected messages.

The simulation can therefore be parallelised by distributing agents across disparate processing nodes and synchronising the message boards to ensure that all agents see the same set of messages. For efficiency, agents are not allowed to read and write to the same board from the same transition function. This avoids the need to synchronise the boards on every single write operation.

The synchronisation of a board is initiated the moment all writes have completed using a message board *Sync Start* function. This function is non-blocking and the synchronisation process is performed in a background thread. The framework is then free to execute other functions that do not depend on the board in question. It is possible for multiple message boards to be synchronised concurrently. Before executing agent functions that reads messages from a board, the message board *Sync Complete* function has to be called. This function checks the status of the synchronisation process and returns immediately if the synchronisation is complete. However, if synchronisation is still in progress the function blocks until completion.

Using this distributed method of synchronisation, agents are able to operate asynchronously to perform computation until a dependence on one of the message boards is required. Only the message boards require synchronisation which is performed through a decentralised mechanism. Using this approach, centralised control of the model is only required during the models initial set-up phase (i.e. the mapping of agents ad the initial configuration of the message boards) and not during the simulation of the model.

The agent definitions are written in a dialect of XML called XMML and are parsed together with the entire model by the *xparser* generating the simulation code.

Based on the model definition, the *xparser* produces a directed acyclic graph representing a dependency graph of transition functions. Each agent model will have its own function dependency graph and they are coupled together by dependencies on message boards. Nodes representing a message board are dependents of functions that write to the board, and dependencies of functions that read from the board.

Using the function dependency graph, the *xparser* can schedule the execution of agent transition functions such that message producers are scheduled as early as possible and message consumers as late as possible. This maximises the amount of computation being performed while the synchronisation process is in flight.

It is possible for transition functions of different agent types to be interleaved as long as the dependencies are met.

Agent instances are represented as a *struct* containing the internal memory of the agent. Agent transition functions read this memory *struct* and updates its values, effectively transitioning the agent instance to the next state ready to be consumed by the next function. The execution of a transition function is repeated for all agent instances of the associated type in the relevant state. Once all functions have been called (in the correct order so as to meet dependencies) an iteration of the simulation is complete.

3.2 Overview of the Design of FLAMEGPU

FLAMEGPU is an agent-based modelling framework that exploits the parallel architecture of the (GPU) offering integrated real time visualisation and model interaction. It builds on the work of FLAME and uses a variant of the XML based model description with a custom XSLT based code generation [37] process, rather than using the FLAME *xparser*.

From a simulation perspective, FLAMEGPU utilises the Single Program Multiple Data (SPMD) architecture of GPUs to map agent functions as GPU kernels operating synchronously over agent and message memory stored as arrays of linearly offset data. Whereas FLAME stores agent memory as Array of Structures (AoS), FLAMEGPU uses Structure of Arrays (SoA) to ensure all memory access is coalesced. During simulation, centralised control of agents is maintained by a host CPU thread which is responsible for scheduling the execution of the GPU kernels. This centralised control has limited impact on performance as data is maintained on the GPU device (avoiding costly data transfers) and CPU control is restricted to determining the order of kernels to be executed.

FLAMEGPU provides a massive performance increase over FLAME [38] but is best suited to large populations of relatively simple (in terms of agent memory requirements) agents. This provides a good balance between the large number of threads required to hide memory access latencies with the limited register availability of the underlying architecture. As FLAMEGPU has no support for multiple GPU devices, models are also restricted to the memory space available on a single GPU device.

3.3 FLAME-II

Some of the FLAMEGPU design solutions and the limitations identified for the FLAME implementation, mentioned below, have led to the design of a new version of the FLAME software platform, called FLAME-II .

3.3.1 Limitations of the FLAME Design

To define an agent, modellers specify a set of state transition functions to transition an agent from one state to another. These functions have read-write access to all variables within agent memory which seems sensible at first, but in hindsight is the cause of (or a contributing factor to) some of the limitations of the design.

Data Granularity. Because each function can potentially write to all memory variables, the smallest unit of data is the whole agent instance. Data partitioning for parallel execution has to therefore be done at the agent level. So, we have to wait until agents have finished an iteration to write their memory to disk. This can be a sizeable proportion of the simulation process time depending on the model and population size.

Execution Path Bound by Model Definition. The memory access requirements of each transition function are not known to the framework. The parser therefore cannot make any assumptions about the actual dependencies between the functions and has to rely solely on the state diagram defined by the modellers. More often than not, this leads to false dependencies between functions and an execution graph that is mostly sequential with very few concurrent paths.

Thread Safety.

Due to the lack of thread safety, the framework is unable to safely execute multiple transition functions concurrently and is therefore less able to efficiently utilise multi-core systems.

3.3.2 FLAME-II Design

The following sections discuss some of the approaches that we have used in FLAME-II to explore and maximise the parallelism potential within agent-based models.

Discovering More Parallelism Through Data Dependency Analysis. To improve the parallel performance of the framework, we need to extract as much concurrency as possible from a simulation. This involves breaking the simulation down into more parallelisable units and then scheduling their execution in a manner which fully utilises all resources available to the execution environment.

Decomposing Agents into Independent Vector Operations. With the changes introduced in the previous section, transition functions can be treated as operations on a predefined set of independent variables. Since all the agents of the same type have the same set of transition functions and memory structure, we can effectively treat the transition function as an operation on long vectors where each vector element corresponds to an agent instance.

Dynamic Task Scheduling. The dependency graph generated based on the analysis of memory reads/writes would implicitly encode the data dependencies. Therefore, as long as the function dependencies are met each function is guaranteed to be accessing the correct versions of memory and messages. This greatly simplifies the job of managing function dependencies and ensuring the correctness of the simulation.

Using Multiple Queues to Manage Different Resources. Assigning a task type allows us the opportunity to support multiple task queues. Each queue can be assigned to different resources that can be managed independently.

4 Benchmark Comparisons

The performance of the current state of the FLAME agent-based software, consisting of FLAME-II running on different hardware and software environments, including HPC platform, and FLAMEGPU running on the parallel architecture of the GPU, will be illustrate through a benchmark problem. This, referred to as the Circles model, consists of only a single agent and message type with three agent functions, which output and input a message containing location information with a final function that moves the agent according to inter-agent repulsive forces [37, 39].

To give a comparison between an early build of FLAME-II and the previous version of FLAME, the Circles model using 50000 agents was used to predict performance. The machine used for benchmarking contains 16 physical cores (Intel® Xeon® CPU E5-2687W 3.10GHz) with 64GB RAM running Ubuntu and using gcc/g++ for compilation. FLAME used round robin partitioning while FLAME-II used vector splitting. The *number of procs* from now on refers to the number of cores (or the number of MPI tasks/processes, which is equivalent).

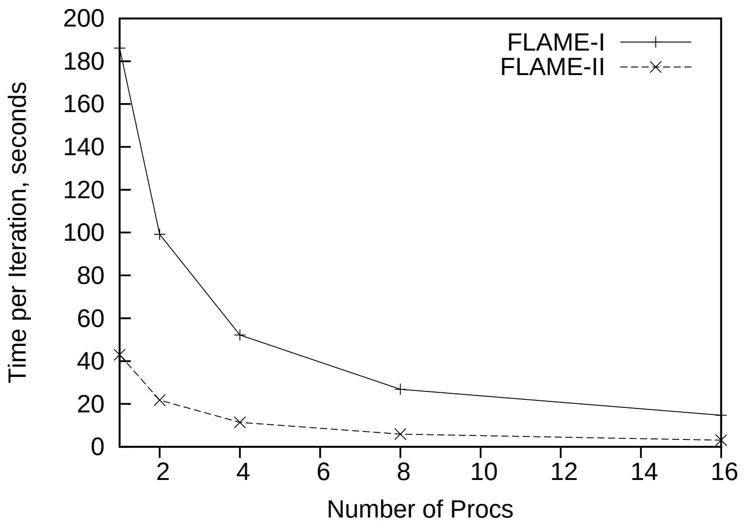

Fig. 1 Comparison of time per iteration between FLAME and FLAME-II

Figure 1 shows the shortest iteration time over multiple runs utilising different number of procs. FLAME-II shows significant speed improvements when compared to FLAME-I.

Figure 2 shows the speedup ratio of parallel runs over the equivalent sequential run. At this early stage it can be shown that the FLAME speedup ratio starts to tail off with a higher number of processors earlier than the FLAME-II speedup ratio.

It can be shown that for this test model FLAME-II scales better with increasing numbers of procs.

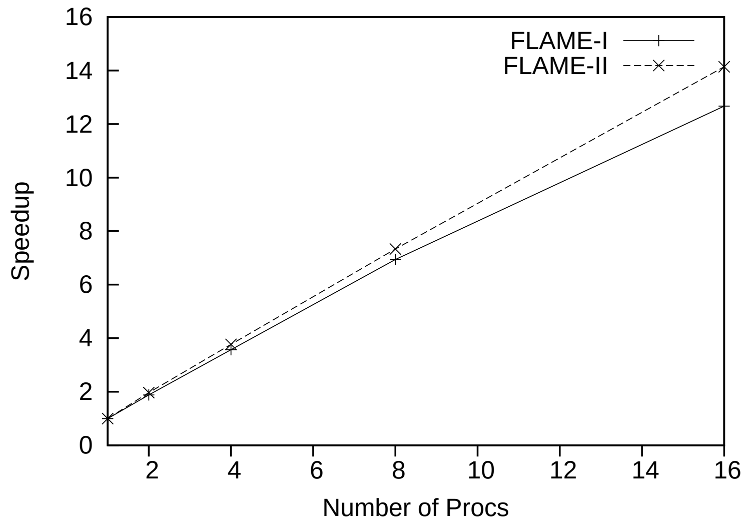

Fig. 2 Comparison of speedup between FLAME and FLAME-II

We will illustrate FLAMEGPU performances by using the same benchmark problem running now on an NVIDIA Tesla K40 GPU platform. The results are presented in Table 1. In these experiments the population size is increased from 1024 agents to 1048567, by doubling in each case the previous population size (column labelled `Pop Size`) and using an environment area (`Env Area`) such that the population density is constant (16). The results in the last two columns are obtained using two different communication algorithms, a Brute Force (`BF Comm` column) and Spatial Partitioning (`SP Comm` column). The performance measurements, in ms, are made by averaging the performance over 10 iterations.

The Brute Force communication algorithm considers that each agent reads the message of every other agent in the system. This is comparable to what happens per distributed node in FLAME. The Spatial Partitioning communication algorithm deals with a spatial data structure used to limit the number of messages read by each agent based upon the communication radius. It is much more like the distribution of agents across processors in FLAME.

In [38, 40] it has been shown the massive increase in performance when a GPU implementation is used, the results in Table 1 shows that the latest generation of GPU hardware demonstrates considerably higher performances.

Table 1 FLAMEGPU results.

Pop Size	`Env Area`	`BF Comm`	`SP Comm`
1024	64	1.992	1.183
2048	128	3.678	1.453
4096	256	6.886	1.455
8192	512	13.772	1.813
16384	1024	52.787	2.472
32768	2048	160.905	3.198
65536	4096	526.300	5.209
131072	8192	1847.192	16.731
262144	16384	7269.673	62.41
524288	32768	29523.372	159.087
1048576	65536		359.345

5 FLAME Applications

In this section we look at a number of specific cases of the use of the FLAME approach, briefly presenting some examples of systems from biology and economics, pedestrian behaviour and work psychology applications. We focus mostly on simulation aspects involving the use of the FLAME and FLAMEGPU environments, but present also some formal verification aspects as well.

5.1 Applications in Biology

We present three models, following [19], at different levels of organisation: starting at molecular level, then continuing with tissue and finishing with an example of organisms in a colony of social insects.

5.1.1 Model of the Innate Immune System - the NF-kB Pathway

In this model [41] each relevant molecule is modelled as an agent which can move around the cell and interacts with other molecules under suitable conditions. These molecular agents diffuse through the cell, binding and dissociating from other molecules, receptors and cell structures – all of them represented as agents – in accordance with the circumstances pertaining at that precise moment. Each agent communicates by sending or receiving signals. Two molecules interact according to certain rules and depending on their state and proximity, derived from standard rate constants. The model then implements the various chemical reactions that take place between these agents. The simulations try to faithfully represent what is known about the type of various molecules, their position inside the cell and the complex series of reactions.

Using single cell data, the model demonstrated remarkable agreement with the experimental data and provided the opportunity to investigate various alternative behaviours. Experimental data indicated that there is a mismatch between the amount of IkB in the cell and the amount needed for the NF-kB pathway. Using the model, we tested a hypothesis, the surplus was sequestered in the actin filaments under normal conditions, and this was then experimentally validated [42].

5.1.2 Oxygen Metabolism in Aerobic-Anaerobic Respiration in *Escherichia coli*

The Bacterium *Escherichia coli* (*E. coli*) is one of biologys key model organisms, probably the best characterized bacterium. *E. coli* is biochemically versatile and unlike many organisms can thrive in environments either with abundant oxygen or no oxygen (O_2) [43]. The ability to sense and respond to changes in O_2 availability is necessary for *E. coli* to successfully compete in a range of niches, including during infection and when used as a cell factory in biotechnology. In order to run experiments for understanding these problems is technically demanding and time consuming, thus accurate models are important. In this model each individual molecule is represented by an autonomous agent. Molecules can move through 3D space in the cell and interact with each other when close enough and in a suitable state. Molecules move differently in different regions of the cytoplasm and this was modelled by controlling the movement of agents in the different areas and changing their location by Brownian motion (random movement of particles suspended in a fluid) where appropriate. Multiple binding of agents are added to chemical reactions, each molecule agent can seek interaction with several relevant types of molecules, with the appropriately sized interaction radius for each type.

The modelling approach in this case consists of a combination of three distinct models: kinetic, reduced-order kinetic, and agent/hybrid modelling. Each of the modelling approaches contributes by addressing questions that are difficult to incorporate within a single modelling framework. The resulting models can provide predictions, be used as a scaffold for our emerging understanding of the data and identify gaps in our biological knowledge. Each component has used different modelling techniques that depend on the availability of biological data. The model has been extensively validated under controlled experimental conditions [44].

5.1.3 Epithelial Tissue and Wound Repair

This modelling approach is at the tissue level where we are investigating how groups of cells interact and form structures and evolve key functions in organisms. We consider interactions between cells in both the bladder and in skin [45, 11, 46, 47, 12].

Some of the key issues related to the growth and repair of tissues are related to the division and migration of cells within a population. Each cell has a fundamental cell cycle that underpins cell growth and division, or provides alternative routes for specialisation (differentiation) and death (apoptosis). Progression around the cell cycle is affected by interactions with other cells, either through direct cell:cell contact or indirectly through the release and detection of soluble signalling factors, which may have a profound effect on behaviour. These are the problems that modelling has to tackle. The agents in this model represent cells that move in space, but also divide, grow and eventually die.

In virtuo investigations indicated that both cell proliferation and migration are crucial for re-epithelialisation, suggesting delicate mechanisms to coordinate the behaviour of different keratinocyte populations. Further model analysis found certain factors playing a positive role in epidermal wound healing by coordinating the behaviour of these keratinocyte populations,.

The problem has been also mapped into FLAMEGPU and various experiments performed [18]. The performance of the GPU version against the CPU shows a great potential in using FLAMEGPU for such problems. The simulation that took several hours using a single CPU core, was processed on the GPU in less than 2 minutes. FLAMEGPU allows the real-time visualisation to be coupled with the simulation, which is a real benefit for real-time interaction during the simulation.

5.1.4 Foraging Strategies in Ant Colonies

We are moving now to modelling the behaviour of ant colonies. A fundamental requirement for insect societies living in a central place is the discovery and efficient exploitation of food sources. We investigated the foraging activities of the Pharaohs ant, Monomorium pharaonis, a small tropical ant species. They organise their foraging activities using chemical communication to produce pheromone trails. Ant pheromone trails provide an effective and efficient solution to the problem of locating and exploiting available food resources.

In this context various ants are modelled as agents participating to foraging activities. A notable discovery facilitated by agent-based modelling is that Pharaohs ants produce trail networks with a treelike structure [48]. The branches of this network have a mean bifurcation angle of 54 degrees, and this branching structure conveys important information to the ants. Simulations of foraging ant colonies prompted a new hypothesis of ant orientation in trail networks which was confirmed by extensive experimental observations [49].

There are a number of key principles that need to be observed if realistic models of biological systems are to be achieved. Biological systems exist in a physical three-dimensional world governed by the laws of physics. There is a strong temptation by computer scientists to abstract away some of these factors in order to make modelling and analysis more tractable. This may impede useful insights into biology since ignoring geometry and the real forces that dictate system behaviour can be very misleading. The advantage of agent-based modelling over traditional mathematical modelling, such as differential equations, is that many of the key determinants underpinning the emergence of complex system behaviour are found in behaviour of individual molecules, cells of organisms. Agent-based modelling enables us to understand the emergent development of structure and function and provides a deep understanding of biology [19].

5.2 Applications in Economy

There are various ways of modelling modern economies and mechanisms for carrying out policy analysis. Mathematical models have been successfully used for many years and they still represent a powerful tool in this respect. However, these tools make sometime gross simplifications and this requires new alternative approaches [50]. New ways of looking at economics more grounded in reality are now developed using agent-based computational models.The idea is not new, but previous attempts in this respect have been limited by the inability to handle large populations of agents. Nowadays, with the increasing computational power and availability of high performance computers some of these limitations have been overcome. In the sequels we present, following largely [50], how FLAME, due to its ability to run simulations on powerful high performance machines, has been used to provide the first detailed description of the entire European economy created using the agent-based modelling technique (this research has been funded through the European EURACE project – www.eurace.org).

The different markets are modelled in great detail. For example, agents in the labour market are considered with firms seeking credit, firms buying capital goods, deciding production levels, advertising and appointing staff, selling consumption goods etc. The full list of agent types is ([50]):

- Firms (consumption goods producers)
- Households (workers and consumers)
- Investment goods producers
- Malls (retail outlets selling consumption goods)
- Banks (providing credit and taking savings and investments)
- Clearing house (managing the market and buying and selling of equity shares, bonds etc.)
- Government (setting fiscal, labour and other policy and collecting taxes, offering subsidies etc.)
- Central Bank (managing money supply, interest rates etc.)
- Eurostat (collects and reports economic statistics etc.)

A number of policy experiments have been carried out to demonstrate the capabilities of the complete model - some are illustrated below.

Experiment 1. An analysis of the impact of fiscal tightening compared with quantitative easing in economic performance [51].

One of the conclusions of this study is that the quantitative easing policy provides better macroeconomic performance than fiscal tightening. This can be explained by the fact that private sector agents, being endowed with higher monetary resources, are more able and willing to consume and invest. The higher monetary endowment is not offset by higher prices. Quantitative easing however does not stimulate more lending from the banking sector, as it has been claimed recently. Indeed, the opposite is true. Quantitative easing, being a lender of last resort mechanism to the government, provides a better macroeconomic performance by reducing the crowding out effect on the demand side of the economy that may be caused by the financing of public debt.

Experiment 2. The effect of open migration within the EU on the economies of the member countries [15].

The simulations indicate that the free migration of workers from a new accession country to an established country brings considerable economic benefits to the receiving country. The down side is that the contributing country suffers an economic penalty. This should be seen in the light of the recent UK policy to free migration from the new EU countries in contrast to the policy of some other countries – e.g. Germany.

Experiment 3. The effects of a policy of household subsidies in dealing with an exogenous oil price shock [17, 16].

The impact of a sudden rise in a key commodity such as oil can be highly damaging to an economy. This can be shown using the model. By have incorporating energy prices into the capital goods prices and using an energy price mark-up one can show how a prolonged energy crisis may affect GDP growth negatively.

The EURACE model built using the FLAME environment has demonstrated that agent-based modelling of large-scale economic systems is both possible, tractable and produces results that provide a good basis for more insights into the behaviour of more complex economic problems.

5.3 Other Applications

In this section we present other applications of the FLAME and FLAMEGPU environments in different areas. We start by presenting an application with impact in social crowd behaviour. This application has also a significant impact on some graphics oriented problems and simulations.

The problem of *pedestrian simulations* have found increasing use in various scenarios and circumstances dealing with highly populated areas, like airports, sport arenas, pilgrimages to religious places. These simulations require flexibility in dealing with various hypotheses, real-time solutions to some problems, easy and effective ways to represent and manipulate agents.

A large scale pedestrian simulation method, implemented with an agent-based modelling approach and running on a FLAMEGPU environment, is presented in [40]. The techniques used for pedestrian simulation make use of parallel processing through graphics cards hardware allowing simulation scales that go far beyond those of serial frameworks. The method allows rapid prototyping and robust validation and testing through the use of a generic abstract framework. The method implemented benefited from the use of an optimised path followed by pedestrian, showing that the optimised flow generates a more realistic flow.

In [52, 53] it is presented a *psychology-based study on teams behaviour* in software engineering projects in certain real-life situations, where different software development methods, like waterfall and agile methods, are used. By using an implementation of the concept of transactive memory in simulation scenarios, it is shown how the knowledge about the skills and abilities of team mates can contribute to a successful approach on task allocation and problem solving with respect to software development projects.

In [54] it is initiated another interesting application for FLAME, the automatic *translation of certain formal models* - in this case kernel P systems - into the FLAME environment. This is an important problem, at least for two reasons: (i) it allows the automatic translation of kernel P systems applications to FLAME and (ii) it provides a simulation environment for these applications that can be compared with those dedicated to kernel P systems models. In [54] it is shown how FLAME can be used as a simulation platform that brings new insights into the understanding of complex situations, in this case the subset sum NP complete problem.

5.4 Formal Verification

In the previous sections the emphasis has been on large scale simulation, flexibility in representing agents, robust design and validation through experiments. In some situations it is necessary to verify various hypothesis and running even multiple experiments is not always enough. Certain formal methods and tools widely used in software engineering and program and software analysis, can be used in order to check various properties. This approach requires some sort of specific formal description of the system and comes with certain constraints regarding the size of the system. The method that we describe briefly below is called model-checking and it allows to check certain properties of a system expressed in a certain formal language. We illustrate this approach for the problem of oxygen metabolism in aerobic and anaerobic respiration in *Escherichia coli* [43].

In [55] it is described the process of transforming the above mentioned problem from its FLAME specification into specific model checker tools. Two model checkers have been chosen, ProB and Spin with their associated formal languages, EventB and Promela, respectively. The translation from FLAME X-machine formalism directly to these formal languages is not performed directly as it leads to the usual problem of state explosion that all the model checkers face when the system is relatively complex. In this respect, a better abstraction of the system has been chosen, by using an intermediate kernel P systems model, a rule-based formalism [54], which allows a more systematic approach on representing the rules involved as either rewriting or communicating transformations. Once these translations have been mapped into EventB and Promela, properties expressed also in a formal language, a specific type of temporal logic called Linear Time Logic (LTL), have been formulated and verified - some examples are provided below. The aim of verifying these properties is to check certain type of behaviour that occurs irrespective of how many simulations are performed.

For example, the interaction mechanism between a Fnr monomer and oxygen molecule is defined as follows in the model considered [55]:

- When an oxygen molecule is within a pre-defined reaction distance to an Fnr dimer, the Fnr dimer is decomposed into two Fnr monomers. If this dimer is bound to a binding site, the binding site will become unoccupied.
- When two Fnr monomers are within reaction distance, they can be combined into an Fnr dimer.
- When the distance between an Fnr dimer and an unoccupied binding site is less than their reaction distance, the dimer will bind to the binding site.

Under certain initial conditions, when the number of oxygen molecules is less than 100 and there are no more than 75 dimers and 18 binding sites, one can show that the number of Fnr dimers is low. This can be shown by using an LTL query, $G(noMon < 7)$, which shows that in any execution pathway (simulation) the number of such monomers is no more than 7. Other more subtle properties can be verified. For instance, given an initial concentration of 500 molecules of oxygen, after 500 steps, the number of FnR monomers will be at most half the number of FnR dimers; whereas when we start with 100 oxygen molecules, after the point when the oxygen disappears from the cell, no Fnr monomers will be produced.

The experiments with two model checkers have shown their abilities in dealing with various aspects of the verification process, as well as their limitations. EventB having a formalism based on the set theory, uses functions, sets and set operators as building blocks for specifying the molecule evolution rules. The non-deterministic conditional and cycling instructions available in Promela recommends it as a suitable specification language for modelling the non-deterministic behaviour of the different molecules. Overall, one can conclude that EventB proved to be more convenient for modelling, while Spin was more efficient for simulation and verification, according to the experiments conducted in [55].

5.5 Potential use within Data Intensive Computing Domains

With respect to the relationship between the compute intensive FLAME and FLAME GPU architectures and more data intensive computing, previous literature [56] has highlighted how agent based modelling can have a two tier impact into knowledge extraction from data. On the one hand data mining techniques can be applied to large scale ABM (such as those produced by FLAME and FLAME GPU) to help analyse simulation data as well as to help in the validation and verification of models through knowledge driven analysis of results. Conversely, ABM can be applied to data mining as a method of generating quasi-real data, where areas of data are missing or incomplete. In this approach the FLAME modelling approach could be used predictively (in much the same way as it has been applied in the case studies presented) to produce data to be combined with exiting or partial data sets such as those prevalent within social or biological sciences. Given that the biological and social sciences are transitioning towards data rich science. The application of large scale agent based modelling frameworks can play a vital role when used in combination with data intensive methods such as data mining. Finally, the potential to use the inherent scheduling and distribution algorithms of FLAME to parallelise large scale data analytics can be achieved through the use of intelligent agents. In this case agents can perform data analysis directly using communication where needed to negotiate.

6 Conclusions

In this chapter, we have described the current design of FLAME and FLAMEGPU. We then described an architecture, FLAME-II, that decomposes the simulation into a list of vector operations that can be scheduled based on a dependency graph. The dependency graph can be generated by analysing the memory accesses of each agent function.

The new approach will enable various optimisation opportunities including more efficient data structures, better resource utilisation using dynamical task scheduling, and multiple levels of parallelism. Each of these was briefly discussed.

The benchmark results provided, while not comprehensive, give a good indication that the performance of an early build of FLAME-II is already significantly beating the performance of the previous version of FLAME. FLAMEGPU is also shown to outperform FLAME on some specific applications, whereas specific optimisation algorithms regarding the communication mechanisms can improve even more its performances.

A set of applications from various fields prove the flexibility, efficiency and usefulness of these software environments. Finally, the issue of formally verifying these systems is discussed for the case of a rule-based system which can be formally analysed using model checking techniques.

Future work includes the development of the FLAME-II platform to the level of fully integrating FLAMEGPU such as to obtain a unique simulation environment that covers a broad range of hardware platforms and provides support for approaching complex problems and running large-scale simulations.

Acknowledgement

This work has been funded by EPSRC Grants EP/I030654/1 and EP/I030301/1 and the University of Sheffield Vice Chancellors Fellowship Scheme.

References

1. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: A multi-agent simulation environment. Simulation: Transactions of the society for Modeling and Simulation International **82**(7) (2005) 517–527
2. North, M., Collier, N., Vos, J.: Experiences creating three implementations of the Repast agent modeling toolkit. ACM Transactions on Modeling and Computer Simulation **16**(1) (January 2006) 1–25
3. Minar, N., Burkhart, R., Langton, C., Askenazi, M.: The Swarm simulation system: a toolkit for building multi-agent simulations. Working Paper 96-06-042, Santa Fe Institute (1996)
4. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL: NetLogo (1999)
5. FLAME Website: http://www.flame.ac.uk (2013)
6. Heath, B., Hill, R., Ciarallo, F.: A survey of agent-based modelling practices. Journal of Artificial Societies and Social Simulation **12** (2009) 9
7. Allan, R.: Survey of agent-based modelling and simulation tools. Technical Report DL-TR-2010-007, Science and Technology Facilities Council (2010)
8. Weidlich, A., Veit, D.: A critical survey of agent-based wholesale electricity. Energy Economics **30** (2008) 1728–1759
9. Leitäo, P.: Agent-based distributed manufacturing control: A state-of-the-art survey. Engineering Applications of Artificial Intelligence **22** (2009) 979–991
10. Friesen, M.R., McLeod, R.D.: A survey of agent-based modelling of hospital environments. IEEE Access **2** (2014) 227–233
11. Sun, T., McMinn, P., Coakley, S., Holcombe, M., Smallwood, R., MacNeil, S.: An integrated systems biology approach to understanding the rules of keratinocyte colony formation. J. R. Soc. Interface **4** (2007) 1077–1092
12. Adra, S., Sun, T., MacNeil, S., Holcombe, M., Smallwood, R.: Development of a three dimensional multiscale computational model of the human epidermis. PLoS ONE **5** (2010)
13. Li, X., Upadhyay, A.K., Bullock, A.J., Dicolandrea, T., Xu, J., Binder, R.L., Robinson, M.K., Finlay, D.R., Mills, K.J., Bascom, C.C., Kelling, C.K., Isfort, R.J., Haycock, J.W., MacNeil, S., Smallwood, R.H.: Skin stem cell hypotheses and long term clone survival – explored using agent-based modelling. Scientific Reports **3** (2013)
14. Burkitt, M., Walker, D., Romano, D., Fazeli, A.: Modelling sperm behaviour in a 3D environment. (2011) 141–149
15. Dawid, H., Gemkow, S., Harting, P., Neugart, M.: On the effects of skill upgrading in the presence of spatial labor market frictions: An agent-based analysis of spatial policy design. Journal of Artificial Societies and Social Simulation **12** (2009) 334–347
16. van der Hoog, S., Deissenberg, C.: Energy shocks and macroeconomic stabilization policies in an agent-based macro model. In Dawid, H., Semmler, W., eds.: Computational Methods in Economic Dynamics. Volume 13 of Dynamic Modeling and Econometrics in Economics and Finance. Springer Berlin Heidelberg (2010) 159–181
17. Deissenberg, C., van der Hoog, S., Dawid, H.: EURACE: a massively parallel agent-based model of the European economy. Applied Mathematics and Computation **204**(2) (October 2008) 541–552
18. Richmond, P., Walker, D., Coakley, S., Romano, D.: High performance cellular level agent-based simulation with FLAME for the GPU. Briefing in Bioinformatics **11** (2010) 334–347
19. Holcombe, M., Adra, S., Bicak, M., Chin, S., Coakley, S., Graham, A., Green, J., Greenough, C., Jackson, D., Kiran, M., MacNeil, S., Maleki-Dizaji, A., McMinn, P., Pogson, M., Poole, R., Qwarnstrom, E., Ratnieks, F., Rolfe, M., Smallwood, R., Sun, T., Worth, D.: Modelling complex biological systems using an agent-based approach. Integrative Biology **4** (2012) 53–64
20. Eilenberg, S.: Automata, languages and machines. Vol. A. Academic Press, London (1974)
21. Holcombe, M.: Towards a formal description of intracellular biochemical organisation. Technical Report CS-86-1, Dept of Computer Science, University of Sheffield, Sheffield, UK (1986)

22. Laycock, G.: The Theory and Practice of Specification Based Software Testing. PhD thesis, Dept of Computer Science, University of Sheffield, Sheffield, UK (1993)
23. Holcombe, M., Ipate, F.: Correct Systems – Building a Business Process Solution. Springer (1998)
24. Barnard, J., Whitworth, J., Woodward, M.: Communicating X-machines. Information and Software Technology **38**(6) (June 1996) 401–407
25. Balanescu, T., Cowling, A., Georgescu, H., Gheorghe, M., Holcombe, M., Vertan, C.: Communicating stream X-machines are no more than X-machines. Journal of Universal Computer Science **5**(9) (1999) 494–507
26. Kefalas, P. Eleftherakis, G., Kehris, E.: Communicating X-machines: a practical approach for formal and modular specification of large systems. Information & Software Technology **45**(5) (2003) 15–30
27. Gheorghe, M., Holcombe, M., Kefalas, P.: Computational models of collective foraging. BioSystems **61** (2001) 133–141
28. Jackson, D., Gheorghe, M., Holcombe, M., Bernardini, F.: An agent-based behavioural model of monomorium pharaonis colonies. In: Proceedings of the 4th International Workshop on Membrane Computing. Volume 2933 of Lecture Notes in Computer Science. (2004) 232–239
29. Holcombe, M., Holcombe, L., Gheorghe, M., Talbot, N.: A hybrid machine model of rice blast fungus, manaporthe grisea. BioSystems **68** (2003) 223–228
30. Coakley, S.: Formal Software Architecture for Agent-Based Modelling in Biology. PhD thesis, Department of Computer Science, University of Sheffield, Sheffield, UK (2007)
31. Sakellariou, I.: Agent based modelling and simulation using state machines. In: 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2012). (2012) 270–279
32. Sakellariou, I.: Turtles as state machines - agent programming in NetLogo using state machines. In: 4th International Conference on Agents and Artificial Intelligence (ICAART 2012. (2012) 235–378
33. Sakellariou, I., Kefalas, P., Stamatopoulou, I.: Evacuation simulation through formal emotional agent based modelling. In: Proceedings of the 6th International Conference on Agents and Artificial Intelligence (ICAART 2014), SciTePress (2014) 193–200
34. Hoops, S., Sahle, S., Gauges, R., Lee, C., Nimus, M., Singhal, M., Xu, L., Mendes, P., Kummer, U.: Copasi – a complex pathway simulator. Bioinformatics **22** (2006) 3067–3074
35. Raymond, G.M., Butterworth, E.A., Bassingthwaighthe, J.B.: JSim: Mathematical modelling for organ systems, tissues, and cells. FASEB J **21** (2007) 736.5
36. Chin, S.: libmboard Reference Manual. 0.2.1 edn. (2009) http://ccpforge.cse.rl.ac.uk/gf/download/frsrelease/107/222/libmboard-0.2.1-UserManual.pdf.
37. Richmond, P., Romano, D.: Template driven agent based modelling and simulation with CUDA. In Hwu, W.m., ed.: GPU Computing Gems Emerald Edition. Morgan Kaufmann (2011) 313–324
38. Richmond, P., Coakley, S., Romano, D.: A high performance agent based modelling framework on graphics card hardware with CUDA (extended abstract). (2009) 1125–1126
39. Coakley, S., Gheorghe, M., Holcombe, M., Chin, S., Worth, D., Greenough, C.: Exploitation of high performance computing in the FLAME agent-based simulation framework. In: Proceedings of the 14th International Conference on High Performance Computing and Communications. (2012) 538–545
40. Karmakharm, T., Richmond, P., Romano, D.: Agent-based large scale simulation of pedestrians with adaptive realistic navigation vector fields. (2010) 67–74
41. Coakley, S., Smallwood, R., Holcombe, M.: From molecules to insect communities – how formal agent-based computational modelling is uncovering new biological facts. Mathematicae Japonicae Online **e-2006** (2006) 765–778
42. Pogson, M., Smallwood, R., Qwarnstrom, E., Holcombe, M.: Formal agent-based modelling of intracellular chemical interactions. BioSystems **85** (2006) 37–45
43. Pogson, M., Holcombe, M., Smallwood, R., Qwarnstrom, E.: Introducing spatial information into predictive NF-kB modelling - An agent-based approach. PLoS ONE **3** (2008) e2367

44. Maleki-Dizaji, S., Rolfe, M., Fisher, P., Holcombe, M.: A systematic approach to understanding bacterial responses to oxygen using Taverna and Webservices. In: Proc 13th International Conference on Biomedical Engineering. (2009) 77–80
45. Walker, D., Wood, S., Southgate, J., Holcombe, M., Smallwood, R.: An integrated agent-mathematical model of the effect of intercellular signalling via the epidermal growth factor receptor on cell proliferation. Journal of Theoretical Biology **242** (2006) 774–789
46. Sun, T., McMinn, P., Holcombe, M., Smallwood, R., MacNeil, S.: Agent based modelling helps in understanding the rules by which fibroblasts support keratinocyte colony formation. PLoS ONE **3** (2008) e2129
47. Sun, T., Adra, S., MacNeil, S., Holcombe, M., Smallwood, R.: Exploring hypotheses of the actions of TGF-$\beta 1$ in epidermal wound healing using a 3d computational multiscale model of the human epidermis. PLoS ONE **4** (2009) e8515
48. Jackson, D.E., Holcombe, M., Ratnieks, F.L.W.: Trail geometry gives polarity to ant foraging networks. Nature **432** (2004) 907–909
49. Jackson, D.E., Martin, S.J., Ratnieks, F.L.W., Holcombe, M.: Spatial and temporal variation in pheromone composition of ant foraging trails. Behavioural Ecology **18** (2007) 444–450
50. Holcombe, M., Coakley, S., Kiran, M., Chin, S., Greenough, C., Worth, D., Cincotti, S., Raberto, M., Teglio, A., Deissenberg, C., van der Hog, S., Dawid, H., Gemkow, S., Harting, P., Neugart, M.: Large-scale modelling of economic systems. Complex Systems **22** (2013) 175–191
51. Raberto, M., Teglio, A., Cincotti, S.: Credit money and macroeconomic instability in the agent-based model and simulator EURACE. Economics, http://www.economics-ejournal.org/economics/discussionpapers/2010-4 (2010)
52. Corbett, A.: Agent-based Modelling of Transactive Memory Systems and Knowledge Processes in Agile versus Traditional Software Development Teams. PhD thesis, Department of Computer Science, University of Sheffield, Sheffield, UK (2012)
53. Corbett, A., Wood, S., Holcombe, M.: It's the people stupid! - Formal models for social interaction in agile software development teams : Advances in Social Sciences Research Journal
54. Bakir, M.E., Ipate, F., Konur, S., Mierla, L., Niculescu, I.: Extended simulation and verification platform for kernel P systems. (2014) 135–152
55. Ţurcanu, A., Mierlă, L., Ipate, F., Ştefănescu, A., Bai, H., Holcombe, M., Coakley, S.: Modelling and analysis of *E. coli* respiratory chain. In Frisco, P., Gheorghe, M., Pérez-Jiménez, M.J., eds.: Applications of Membrane Computing in Systems and Synthetic Biology. Volume 7 of Emergence, Complexity and Computation. Springer Berlin Heidelberg (2014) 247–267
56. Baqueiro, O., Wang, Y.J., McBurney, P., Coenen, F.: Integrating data mining and agent based modeling and simulation. In: Advances in Data Mining. Applications and Theoretical Aspects. Springer (2009) 220–231