FLAME: A PLATFORM FOR HIGH PERFORMANCE COMPUTING OF COMPLEX SYSTEMS, APPLIED FOR THREE CASE STUDIES*

Mariam Kiran, Mesude Bicak, Saeedeh Maleki-Dizaji
Mike Holcombe

Department of Computer Science, University of Sheffield, Sheffield, UK

(Received March 18, 2011)

FLAME allows complex models to be automatically parallelised on High Performance Computing (HPC) grids enabling large number of agents to be simulated over short periods of time. Modellers are hindered by complexities of porting models on parallel platforms and time taken to run large simulations on a single machine, which FLAME overcomes. Three case studies from different disciplines were modelled using FLAME, and are presented along with their performance results on a grid.

DOI:10.5506/APhysPolBSupp.4.201
PACS numbers: 07.05.Tp

1. Introduction

The ideologies surrounding cellular automata models gave birth to the concepts of agent-based modelling (ABM) methods. Introduced by Reynolds in 1985, agent-based models have recently become the driving force in various research areas, especially after the advent of more powerful parallel computers. These allow simulations of large populations of agents to be executed in controlled environments, examining the affects of various rules of interactions among the agents. Agent-based models encourage bottom-up approaches allowing the research to focus on the individual elements interacting with each other rather than looking at the complete scenario as a whole. Initially, the pattern in models was proved using differential equations with more common examples being found in economic modelling, where mathematical formulas are still being used to prove the behaviour of ideas. [1] and [2] have favoured agent-based approaches by saying that

* Presented at the 2nd Summer Solstice International Conference on Discrete Models of Complex Systems, Nancy, France, June 16–18, 2010.

research should be intensified to focus into agents rather than the whole systems, realistically allowing humans to be modelled as agents rather than differential equations.

The model starts with a description about the individual elements in the system which will be represented as agents. The agents are given a set of memory variables, functions and communication protocols which allows them to communicate with each other and the environment. Agents are implemented as separate pieces of code which communicate with each other through communication protocols, also known as messages. Their individual interactions allow certain macro variables to emerge in the system which depicts how the whole system collectively behaves. The simulated model is tested against real data to check if it is an accurate depiction of the system before it can be used further for research purposes.

FLAME (Flexible Large-scale Agent-Based Modelling Environment) is an agent-based modelling environment which allows modellers from all disciplines, to easily write their own agent-based models. It enables various levels of complexity from modelling molecules to complete communities, by only varying the agent definitions and functions [3]. Formal X-machines [4] are used as the agent architecture, which brought in structure, memory, states and transition functions to the agent. The X-machine agents communicate through messages using the interaction rules specified by the modeller. These rules involve posting and reading messages from the message boards. Using a distributed memory model, Single Program Multiple Data (SPMD), the framework handles deadlocks through synchronisation points, which ensures that all the data is coordinated among agents. FLAME stands out from the other agent-based modelling frameworks as it allows deployment of simulations on large parallel computers. This allowed to run large simulations (as many as 500,000 agents) in a matter of minutes enhancing research in terms of time and complexity of models [5]. Over the years various platforms have been released for ABM building each using different programming languages and having their own characteristics. [6] and [7] have provided a detailed comparison between various platforms by implementing similar models on the different platforms.

2. FLAME in detail

FLAME is based on the 'logical communicating extended finite state machine theory' (X-machine) which gives agents more power to enable writing of complex models. Agents are modelled as communicating X-machines, which allow them to communicate through messages as specified in the model (XML) file. Each agent would thus possess the following characteristics:

(i) a finite set of internal states of the agent, *(ii)* a set of transition functions that operate between the states, *(iii)* an internal memory set of the agent, and *(iv)* a language for sending and receiving messages among agents.

Internal states of the agent are different from internal state of the memory of the agent. Internal state depicts a state after or before a function is performed whereas a state of the memory represents the values in memory at that time step.

Modellers need to prepare three files, which are highlighted in Fig. 1 and described briefly as follows:

`Model.xml`: Multiple xml files contain the model structure, such as agent descriptions, memory variables, function names or interfaces, messages.

`Functions.c`: Multiple '.c' files contain the implementations of the agent functions specified in the xml files.

`0.xml`: This contains the initial states of the memory variables of the agents such as the initialisation of all parameters.

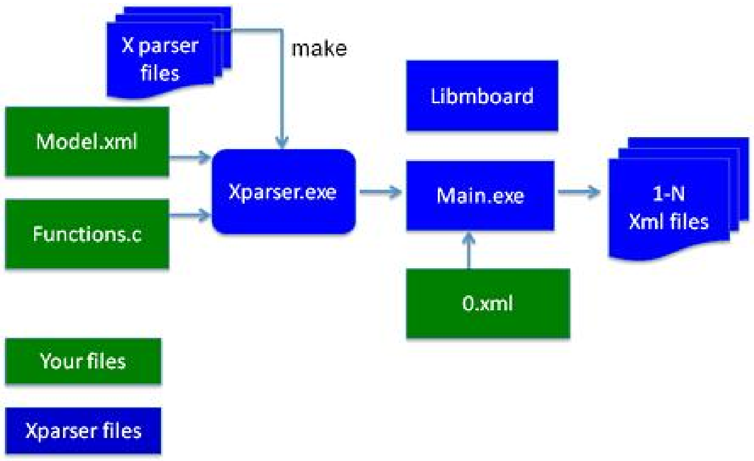


Fig. 1. Block diagram of the FLAME framework.

Xparser, the simulation program generator of FLAME, is compiled with the modellers' files to produce a simulation package (`Main.exe`) for running the simulations. The number of the resulting XML files depends on the number of iterations specified to run a model.

Figure 2 shows the structure of how two X-machines communicate. Communication between agents is handled by an intelligent message board library, which also allows filtering of messages, reducing the work for the agents, thereby improving simulation performances. Conventional state machines have been used to describe the state-dependent behaviour of a system by outlining the inputs to the system, but this failed to include the effect of messages being read and the changes in the memory values of the machine. X-machines are an extension to conventional state machines that include the manipulation of memory as part of the system behaviour, and thus are a suitable way to specify agents. Therefore, describing a system in FLAME includes the following stages: (1) identify the agents and their functions, (2) identify the states which impose some order of function execution within the agent, (3) identify the input and output messages of each function (including possible filters on inputs), and (4) identify the memory as a set of variables accessed by functions (including possible conditions on variables for the functions to occur).

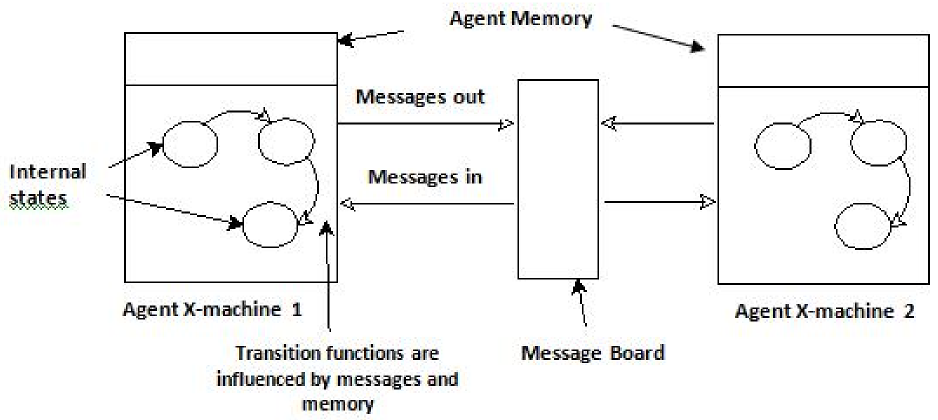


Fig. 2. Communication between two X-machines in FLAME framework.

FLAME also enables models to be parallelised efficiently over parallel computers. Message Passing Interface (MPI) is used to send messages between agents which are located on different processors on various platforms. FLAME reads in the model files and automatically generates a simulation program in C which can be run in parallel by using the '-p' flag or in serial by default. Various parallel platforms like SCARF, HAPU and Iceberg have been used in the development process to test the efficiency of the FLAME framework. Furthermore, FLAME enables agents to be created dynamically throughout a simulation, which is an extremely useful feature for modellers.

3. Case study 1 — economic model

The Cournot model (1897) is a simple economic model which involves firms competing against each other in terms of quantities they produce to achieve the highest profit. The firms produce one homogeneous product and based on the demand in the system the price of the product changes. This is similar to the supply-demand curve in economics literature where the curves have an inverse relationship with each other. The point at which the two curves intersect each other is the equilibrium of the system. At equilibrium the demand would allow the product to cost the optimum market price. Equilibrium in a market scenario is defined as ‘a situation in which the plans of buyers and the plans of sellers exactly mesh, causing the quantity supplied to equal the quantity demanded at the price in the market-place for the good (product)’ [8].

The Cournot model is similarly based on sets of equations that can be used to predict the behaviour of the firms at different times and the equilibrium point for the system. The profit earned by the firm can be calculated from the quantity q_i produced by firm i:

$$\text{Profit}_i = (\text{Pmarket} \times q_i) - (\text{cost}_i \times q_i) .$$

If Firm 1’s production is zero, Firm 2 can dominate the market by producing quantity equal to the demand. When Firm 1 starts producing, Firm 2 should reduce its output as total quantity being produced becomes more than demand. If there is too much of the product, this reduces its sales and Firm 2 will suffer high loss. Note that all products have to be sold in the same iteration. If the total quantity produced is more than demand in the system, product is given away for free or on negative market prices.

3.1. Results and discussion

Table I lists the initial values set at the beginning of the experiment. Figure 3 (a) and (b) display the graphs for the price and profits for the firms in this experiment. The graphs show that the firms take longer to find the equilibrium but eventually oscillate about it. Therefore firms are able to experiment and find an equilibrium among themselves eventually because they are all progressively trying to do better.

TABLE I

Initial numerical values set in the Cournot experiment.

Variable	Value	
QMAX	511	(Assuming all bits in a 9 digit binary string was 1)
n	3	(Number of firms)
Q*	127.5	(Quantity at equilibrium)
P*	128.5	(Price at equilibrium)
Profit*	15108.75	(Profit at equilibrium)

The steps taken during the Cournot model are as follows:
1: Firm Agent: If beginning of the simulation, generate strategies for the firm database else do nothing.
2: Firm Agent: Select an elitist strategy from the memory database based on roulette wheel selection. Post this as the chosen strategy.
3: Firm Agent: Read in all the representative strategies of other firms and calculate the profit I would attain if others played their representative strategies and I played each of the different strategies in my memory. Calculate the new fitness of the strategies accordingly.
4: Firm Agent: Choose two elitist strategies from the database using the new fitness of the strategies. Perform crossover and mutation techniques and find three child strategies.
5: Firm Agent: Use child strategy in the scenario as played strategy.
6: Demand Price Agent: Read in all played strategies by firms and calculate the price of the product depending on the demand in the system.
7: Firm Agent: Reads in the price of the product and calculates the actual profit as a result of playing the child strategy.

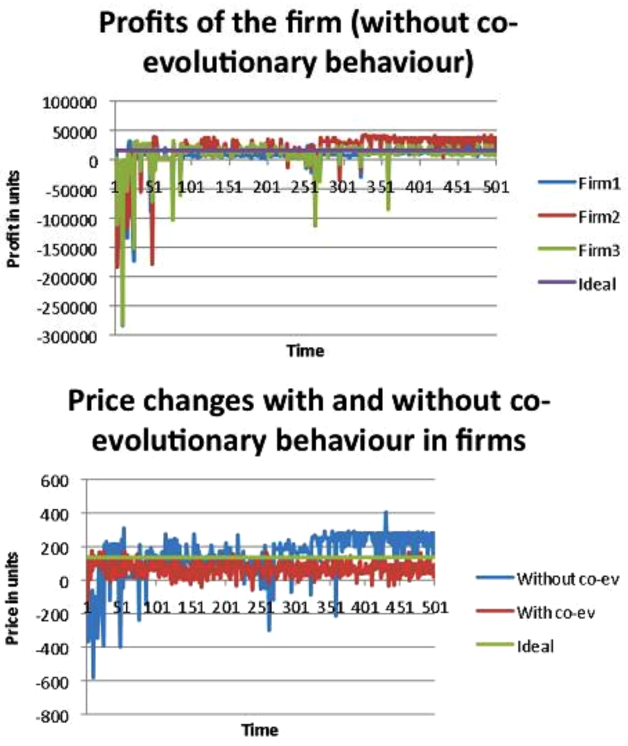


Fig. 3. Graph of profits of firms and price in system.

The Cournot experiment was run on a number of architectures to measure the characteristics of the experiment using the FLAME framework. Figure 4 shows how the simulation times changed when the same model was run on different architectures. The Cournot experiment run with 61 agents, with 3 firms and 1 demand price agent in 20 scenarios and an Averager agent. The figure shows that the model was quicker on the parallel mainframe Iceberg than desktop machines running Windows and Mac.

The figure also shows that the simulation time increased with the number of nodes on the architecture. This increase in simulation time is because the Cournot model is a centralised model and the agents communicated to one demand price agent. Therefore the overhead when all agents send message to one agent increases with nodes. The parallel distribution of the agents was done in a round robin manner distributing the agents uniformly over the processors used. This increased the communication between the different processors which caused a delay in the simulation run.

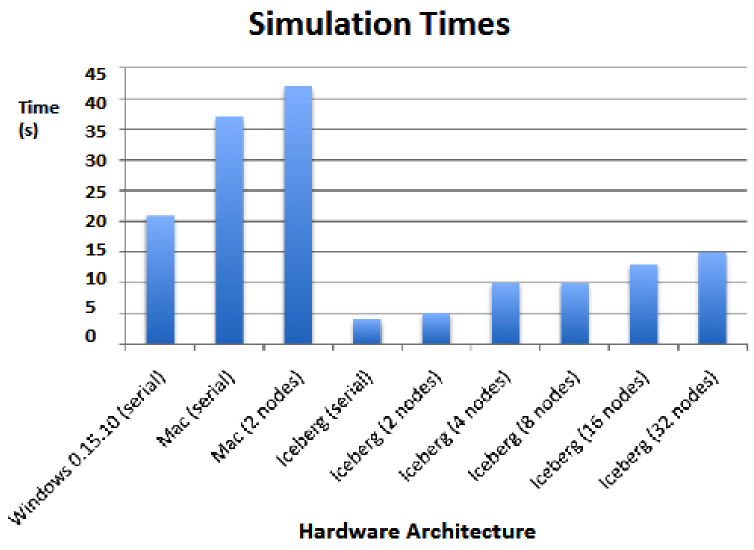


Fig. 4. Simulation times when Cournot model was run on different platforms.

4. Case study 2 — foraging model of *Monomorium pharaonis*

Ant colonies are complex biological systems that respond to changing conditions in nature by solving dynamic problems. Their ability of decentralized decision-making and their self-organized trail systems have inspired computer scientists since 1990s, and consequently initiated a class of heuristic search algorithms [9].

The major challenge in social insect research is understanding how colony-level behaviour emerges from individual interactions. Models to date focus on simple pheromone usage with mathematically devised behaviour, which deviates largely from the real ant behaviour. Furthermore, simulating large-scale behaviour at the individual level is a difficult computational challenge; hence models fail to simulate realistic colony sizes and dimensions for foraging environments.

We work with the Pharaoh's ants, *Monomorium pharaonis*, a widely studied model system for investigating pheromone trails [10, 11]. Pharaoh's ant is a 2 mm monomorphic pest ant forming colonies with less than 2500 workers. They do not have nest-mate recognition, which makes the manipulation of the colony size simple. Having poor vision, they are wholly reliant on pheromones for orientation [10]. Unlike other ant species they deposit trail pheromones constitutively when outside the nest forming branching networks of pheromone trails even before food is discovered [11]. While the majority of ant species (such as *Lasius niger*) form highly diffusive short-lived trails, *M. pharaonis* forms complex persistent networks of narrow trails.

Agent-based modelling approach via FLAME was utilized to prepare a basic foraging model of the Pharaoh's ants. Model rules and parameters were based on related biological research. The aim was to benefit from the advanced features of FLAME and improve the existing ant foraging models in the literature, as well as to test the effect of simple rules introduced to the model.

4.1. Model details

The model consists of ant, pheromone, food and nest agents, as well as two environment agents used for dynamic creation of the pheromone and food agents throughout the simulation. Ant agents are characterised by their identity number, nutritional status, current heading and environmental location. Ant agents exit the nest when their nutrition level drops below a threshold value and begin searching for food. They move non-randomly, according to a probabilistic 'turning kernel' [12], which was empirically derived from biological experiments through video tracking. Ant agents deposit (fed ants deposit 2 units, while unfed ants deposit 1 unit) pheromone agents in the environment as they walk, either creating new ones dynamically or reinforcing existing ones. They have a priority to follow pheromone agents if they can sense any. As observed in biological experiments, ant agents show high fidelity to trail following, but there is always a probability that they will depart from the trail at each step [11]. Agents also engage in U-turning, where they make a spontaneous 180 degree turn [13].

For purposes of simulation colonies of 50, 250 and 500 agents were used. Pharaoh's ant colonies are approximately equally divided into nest-workers and foragers [3], therefore a realistic colony size was simulated. Environment is modelled as continuous space, where each ant step has a length of 2 mm (one ant body length). Dimension of the environment was 500 by 500 ant steps, thus simulating a realistic foraging space of 100 cm^2 [13]. The nest agent has dimensions of 20 by 20 mm, initially containing ant agents with random coordinates inside. Food agents are placed at random coordinates with random sizes, where the size is decreased by 0.02 units whenever an ant agent comes across. When the size of a food agent equals to 0.2 units, the food agent is killed, and a new food agent with random coordinates and size appears, simulating a dynamic foraging environment.

Pheromone agents are characterised by environmental location, concentration and radius. The concentration decays exponentially based on an empirically derived decay rate [11]. The radius simulates the diffusion of the pheromone scent, however due to limited knowledge this is updated based on the concentration via a linear formula. Initially, there are no existing pheromone agents within the environment, simulating a virgin territory. Pheromone agents are dynamically created throughout the simulation by ant agents.

Ant agents search the environment until they detect a food agent, in which case their nutritional level is updated to a maximum food level, and the agent makes a 180 degree turn. Then the ant agent follows pheromone trails to return to the nest. Once an ant agent is back in the nest, she will not leave until her nutrition level falls below the threshold. Their nutrition levels are decreased at every step.

4.2. Results and discussion

Version 1: Basic version based on rules explained in previous section (except probability to leave trail).
Version 2: Version 1 was improved with the addition of a radius to food agents based on food size, representing diffusion of food smell.
Version 3: Version 2 was further modified with a probability to leave the nest (0.001 per step [11]).

Figure 5 shows the results from the three versions of the models. As the environment is simulated as continuous space, the chances of the ant agents missing a food source nearby is very high. Therefore, in version 2 introducing food smell based on food size provided a ground for faster discovery of food agents, as well as establishing stronger trails. Version 3 performed better compared to version 1 as the probability to leave the trail provided the ants with a flexibility in walking off trails which could have been leading the wrong way (exploratory trails), however did not perform as good as version 2, as this also meant that ant agents walked off rewarding trails and got lost in the environment. Furthermore, increased number of ant agents increased the overall foraging efficiency as this meant at any time step, more ants were actively exploring the environment.

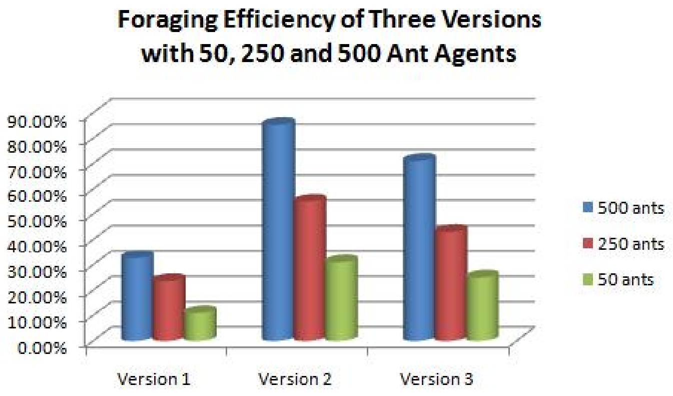


Fig. 5. Foraging efficiency (number of Fed ants/Total number of ants) results from three versions of the model with 50, 250 and 500 ants. Each model was run 10 times for a period of 5000 iterations.

Simulations were performed on a Sony laptop (2.4 GHz processor, 4GB RAM) and Iceberg, the White Rose Grid (586 cores, 435 GFLOPs operating with Sun Grid Engine). Simulations on Iceberg were performed using 1, 2, 4, 8, 16 and 32 cores. Results in Fig. 6 demonstrate that the processing times were reduced by six times when the model was run in parallel compared to serial on the grid, and twenty times when run in parallel on the grid compared to serial on the laptop, which is extremely desirable for processing complex biological models.

FLAME coupled with High Performance Computing enabled large-scale simulations of complex models to be run in parallel on a grid without compromising on the time taken to attain results. In ABM, models can scale up exponentially depending on the number of agents and the complexity of the functions to be performed, therefore this is a significant contribution. Furthermore, the advanced features of the framework, such as dynamic creation of agents during a simulation, provided realistic grounds for modelling agents, especially pheromones, which sets this basic foraging model apart from the related models in the literature.

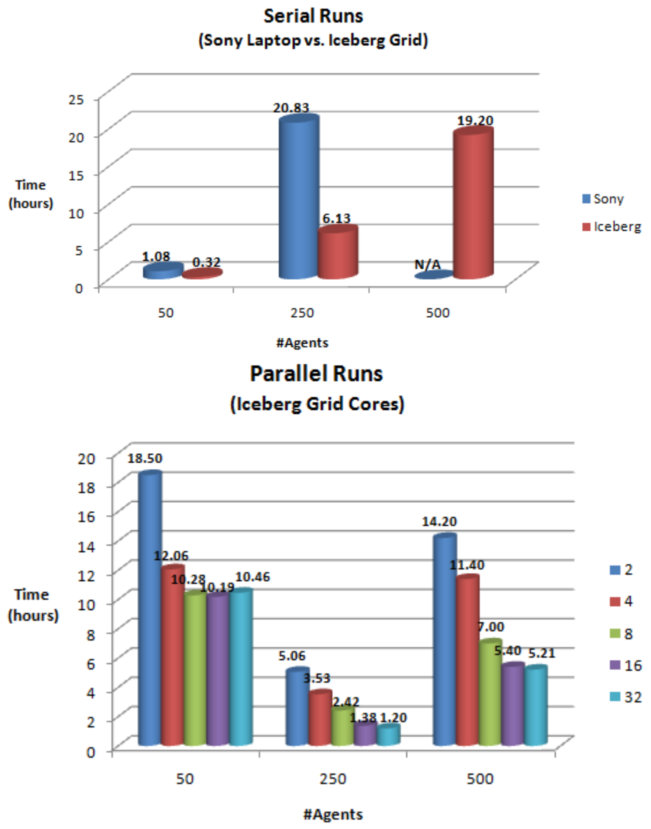


Fig. 6. Performance results from simulations of model Version 2 in serial and parallel on Iceberg grid and Sony laptop [14].

5. Case study 3 — regulation of electron transport chain in *Escherichia coli*

The bacterium *Escherichia coli* is probably the best characterized bacterium as it has been studied for many years. It is relatively easy for biologists to use and probably the most genetically and metabolically defined organism known. Most strains are not pathogenic to humans (they exist in gut flora) but a few cause serious disease *e.g. E. coli* O157:H7. It is biochemically versatile and unlike many organisms can thrive in environments either with abundant oxygen (O_2) or no O_2 [15]. 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. Experimental work with *E. coli* involves the use of Chemostats, which allow measurements of transient properties (metabolite levels, fluxes, H+/O ratios, Q_{O_2}) to be performed in standardized ways that can be replicated in different laboratories. This is the focus of the Systems Understanding of Microbial Oxygen (SUMO) project, a multi-national project in the Systems Biology of MicroOrganisms (SysMO) initiative. Oxygen availability profoundly affects *E. coli* bioenergetics, and through the synthesis of alternative electron transport chains *E. coli* exploits any available O_2 to support aerobic respiration — the most energetically efficient mode of growth. Thus, *E. coli* has two well-characterized alternative terminal oxidases; Cyd has a very high affinity for O_2 and is used under micro-aerobic conditions, whereas Cyo has a relatively lower affinity for O_2 and is used under normoxic conditions (Fig. 7). The synthesis of these alternative oxidases is regulated by two main transcription factors: the fumarate and nitrate reduction regulator (FNR) and the two-component aerobic respiratory control (ArcBA) system (reviewed in [16]). These regulators can sense changes in O_2 availability — FNR is a direct O_2 sensor, whereas ArcBA senses O_2 indirectly [17] — to re-program gene expression such that the most appropriate electron transport chain is synthesized in any particular niche. Hence, in the absence of O_2, expression of Cyo is repressed by FNR and phosphorylated ArcA (ArcA∼P), whereas expression of Cyd is also repressed by FNR but activated by ArcA∼P (Fig. 7). Furthermore, the primary signal for switching ArcBA and FNR off (O_2) is consumed by the terminal oxidases, forming a negative feedback loop (Fig. 7). Thus, the activities of FNR and Arc are the primary determinants of the extent to which each oxidase is synthesized, but measuring these activities experimentally is technically demanding.

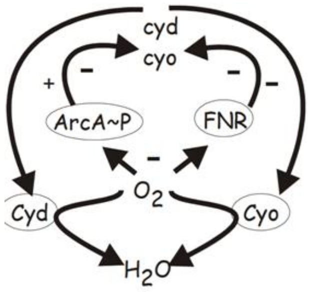


Fig. 7. Schematic model of oxidase regulation.

5.1. Model details

Understanding the synthesis of alternative oxidases in *E. coli* can only be efficiently understood using multi-scale and multi-level modelling. We have integrated three different, complementary modelling approaches: kinetic, reduced-order kinetic, and agent/hybrid modelling (Fig. 8). Each approach contributes by addressing questions that are difficult to incorporate within a single modelling framework. The flexible agent-based supercomputing framework FLAME provides the basis for this integration.

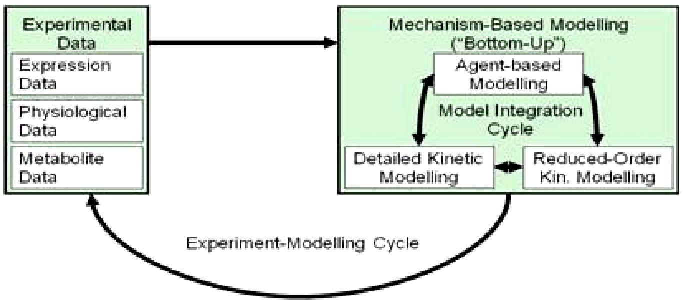
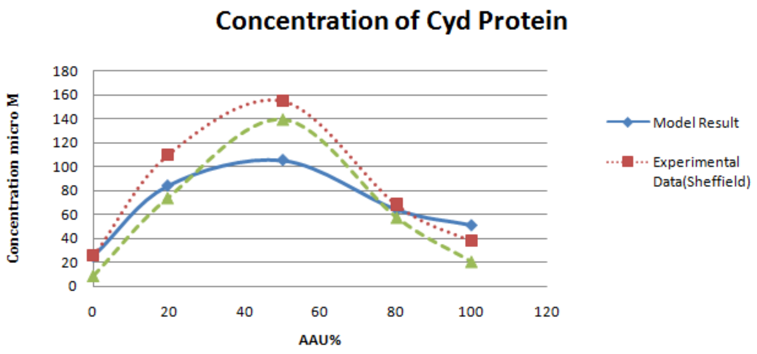

Experimental Data
Expression Data
Physiological Data
Metabolite Data
Mechanism-Based Modelling ("Bottom-Up")
Agent-based Modelling
Model Integration Cycle
Detailed Kinetic Modelling
Reduced-Order Kin. Modelling
Experiment-Modelling Cycle

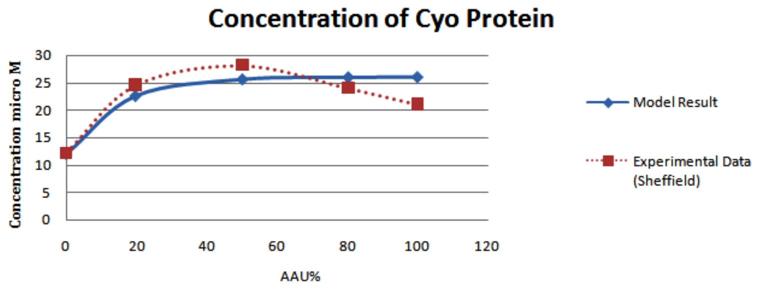
Fig. 8. Experiment-modelling cycle.

Each individual molecule is represented as an autonomous agent that exists within the cellular environment and interacts with other molecules according to the biochemical situation. The numbers of such molecules in a typical cell are between 5000 and 10000 for all of the proteins of interest. Molecules each have a location within the cell. Some, such as the oxidases, are located at the cell membrane, whereas others, like the regulators, are more uniformly distributed. Molecules can move through 3D space in the cell and interact with each other when close enough and in a suitable state. They move differently in different regions of the cytoplasm, which was modelled via Brownian motion where appropriate. The agent-based model must of course agree with the corresponding reaction kinetics model in the circumstance where reaction kinetics can reasonably be applied (*i.e.* with large numbers of molecules of well-mixed chemicals). Since information about reacting chemicals is invariably given for such a situation, and because little information exists about individual molecular interactions, it is important to infer required parameter values from reaction kinetics. Interactions with O_2 molecules were modelled by interpreting the k rate (k is a reaction rate constant that quantifies the speed of a chemical reaction) in terms of interaction radius. The rate constant was then used to deduce information on local interactions. Dissociation was dealt with by applying a probabilistic k rate. Dissociation can easily be accounted for by making bound products separate randomly at a rate specified by the dissociation rate constant (using a uniform random distribution initially, though this could be modified as appropriate). Although this is straightforward, it may be an unnecessary consideration since the rate of dissociation is often negligible. Both k rates have been collected from literature and experimental biologists.

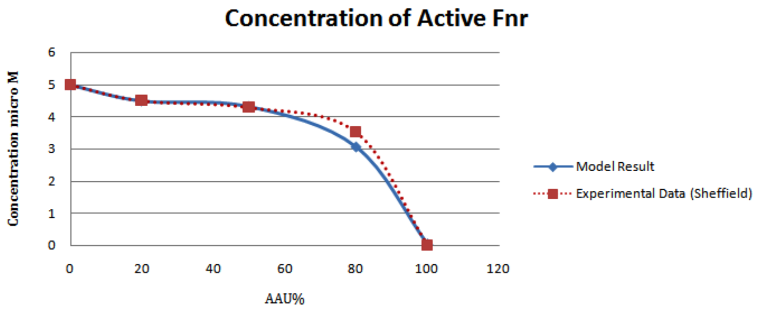
(a)



(b)



(c)



(d)

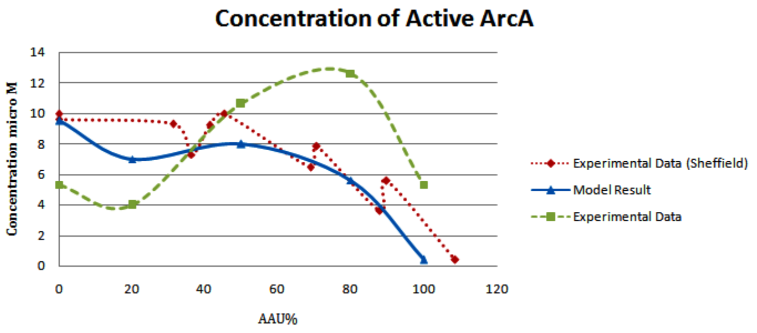


Fig. 9. A comparison between experimental results and the model predictions. Figure (c) shows the disparity between the model predictions and the original experimental data [18].

The capability of multiple agents to bind together was added. This allows us to deal with several chemical reactions, where each molecule agent can seek interaction with several types of molecules, within the appropriately sized interaction radius. Three experimental measurements formed the basis of the model viz. the abundances of the oxidases Cyd and Cyo, the relative activity of the FNR protein in *E. coli* cultures grown under conditions of different O_2 availability. Using these inputs the model was able to closely match the experimental measurements and predict the activity of ArcA in the system (Fig. 9).

5.2. Results and discussion

The predicted ArcA activity did not match ArcA activity predicted from measurement of transcription from an ArcA-regulated promoter. One possibility was that a third O_2-responsive transcription factor might operate alongside FNR and ArcA to regulate both oxidises. Alternatively, the indirect nature of measuring ArcA activity using a reporter fusion could lead to inaccuracies. Therefore, new experimental data directly measuring ArcA phosphorylation was obtained and was found to match the model predictions very closely. Thus the simulations contributed to clarifying and correcting the current knowledge about the role of this important regulatory circuit.

Some of the key ways in which the FLAME-SUMO model has been used is in the estimation of important kinetic parameters in these reactions. Rate constants and concentrations cannot always be measured directly and the agent-based model provides a vehicle for experimenting with these values and then comparing the results with the experimental data derived from the chemostat experiments. The agent-based models have also been used to estimate the level of resources needed by the colonies in order to reach a steady state in a way that may not be possible with other types of modelling.

6. Conclusion and future work

The FLAME framework is being used in a variety of disciplines as presented above. It is a flexible agent based modelling environment which allows production of automatically parallelisable code to run on large mainframe computers. The results presented above have demonstrated how this can be achieved. Future work of the framework target increased efficiency and simulation capacities, where the proposed solutions are the introduction of HDF5 libraries for data compression and transfer over various nodes, and methods to reduce parallelisation bottlenecks by allowing agent migration over nodes to increase simulation speed. The framework can be obtained from `http://www.flame.ac.uk`

Authors would like to thank colleagues of the EURACE project and EU FP6 framework for funding the FLAME research and acknowledge the collaborative work done with STFC at Rutherford Appleton Laboratories on parallelisation of agents in FLAME. M.B. would like to thank Dr Duncan Jackson for his guidance on Pharaoh's ant behaviour and BT Exact for funding PhD research. S.M.D. thanks SUMO Consortium.

REFERENCES

[1] J.H. Miller, S.E. Page, *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*, Princeton University Press, 2007.

[2] J. Epstein, *Generative Social Science. Studies in Agent-Based Computational Modelling*, Princeton University Press, 2007.

[3] S. Coakley, R. Smallwood, M. Holcombe, *Scientiae Mathematicae Japonicae* **64**, 185 (2006).

[4] G. Eleftherakis, P. Kefalas, A. Sotiriadou, E. Kehris, "Modeling Biology Inspired Reactive Agents Using X-machines", in Proceedings of the International Conference on Computational Intelligence (ICCI04), Istanbul, December 2004.

[5] M. Kiran *et al.*, *Biosystems* **93**, 141 (2008).

[6] J. Xavier, K. Foster, *Proc. Natl. Acad. Sci.* **104**, 876 (2007).

[7] S. Railsback, S. Lytinen, S. Jackson, *Simulation* **82**, 609 (2008) [online: `http://sim.sagepub.com/cgi/content/abstract/82/9/609`].

[8] P. Maunder, D. Myers, N. Wall, R.L. Miller, *Economics Explained: A Course-book in a Level Economics*, Collins Educational, London, UK, 1989.

[9] M. Dorigo, Optimization, Learning and Natural Algorithms (in Italian), PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[10] V. Fourcassie, J.L. Deneubourg, *J. Physiol. Entomology* **19**, 291 (1994).

[11] D.E. Jackson, S.J. Martin, M. Holcombe, F.L.W. Ratnieks, *Animal Behaviour* **71**, 351 (2006).

[12] A.D. Vincent, M.R. Myerscough, *J. Math. Biol.* **49**, 391 (2004).

[13] D.E. Jackson, M. Bicak, M. Holcombe, "Decentralized Communication, Trail Connectivity and Emergent Benefits of Ant Trail Networks", Online Journal of Memetic Computing, 2010.

[14] M. Bicak, M. Kiran, "Simulating Decentralised Behaviour of Ant Colonies", Presented at All Hands Meeting 2009, Oxford, UK, 2009.

[15] J.R. Guest, J. Green, A.S. Irvine, S. Spiro, *The FNR Modulon and FNR Regulated Gene Expression*, in E.C.C. Lin, A.S. Lynch (eds.) *Regulation of Gene Expression in Escherichia coli*, R.G. Landes & Co., Austin, Texas, 1996, pp. 317–342.

[16] J. Green, M.S. Paget, *Nature Reviews Microbiology* **2**, 954 (2004).

[17] D. Georgellis, O. Kwon, E.C. Lin, *Science* **292**, 2314 (2001).

[18] S. Maleki-Dizaji *et al.*, *Online Journal of Bioinformatics* **1**, 51 (2009).