20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

A High Performance, Agent-Based Simulation of Old World Screwworm Fly Lifecycle and Dispersal using a Graphics Processing Unit (GPU) Platform

Mitchell Welch[a], Paul Kwan[a], A.S.M. Sajeev[a]

[a] *School of Science and Technology, University of New England, Armidale NSW 2351, Australia.*
Email: mwelch4@myune.edu.au

Abstract: The Old World Screwworm Fly (OWSWF), *Chrysomya bezziana*, is an insect pest that is endemic to the tropical regions of Asia, the Middle East and Africa. The insect reproduces by laying its eggs in open wounds and mucus membranes of warm blooded mammals. Upon the hatching, the OWSWF larvae eat the living flesh of the host animal, causing injury, secondary infections and in extreme cases death. If this pest was introduced to the Australian mainland, it could have a devastating impact on the livestock industries within the northern regions of Australia. This work builds upon the existing research surrounding the OWSWF biological lifecycle and dispersal characteristics by developing a national-scale, high-resolution, agent-based model capable of simulating an invasion of Australia by the OWSWF. The challenge in designing large scale high-resolution models to run on personal computers is addressing performance issues.

We face this challenge by making use of *Graphics Processing Unit (GPU)* technologies*,* based around NVidia's *Compute Unified Device Architecture (CUDA)*, to simulate the lifecycle and dispersal of the OWSWF at the individual insect and cohort levels. This model combines agent-based logic, for simulating the OWSWF's lifecycle, with an efficient cellular-automata system to capture the spatial aspects of the OWSWF population's dispersal during a simulated invasion. The lifecycle and dispersal simulation is supported by an efficient system of main memory management which integrates bio-climatic data from a standard database management system for use within the model. The scheme adopted breaks this agent logic down into GPU-based functions, known as *kernels,* and uses the well-developed heterogeneous programming approach to distribute processing tasks between the *Central Processing Unit* (CPU) of the host machine and the CUDA device.

Analysis of the performance of the CUDA implementation reveals significant improvement in execution time when compared to an equivalent CPU-only based implementation, with results showing that the CUDA implementation's processing efficiency scales up well as the number of agents within the simulation increase.

Keywords: *Agent-Based Model, Graphics Processing Unit (GPU), Old World Screwworm Fly (Chrysomya bezziana), Lifecycle and Dispersal Simulation*

1. INTRODUCTION

In recent years, agent-based modelling has emerged as a promising approach for simulating complex systems across numerous domains, and for a wide range of purposes including social science, ecology, biology and epidemiology (Berger, 2001; Bonabeau, 2002; Bone & Dragicevic, 2009; Connell, Dawson, & Skvortsov, 2009; Schelhorn, O'Sullivan, Haklay, & Thurstain-Goodwin, 1999). Agent-based models are developed around the principle of conceptually breaking complex systems down into individual components referred to as *Agents. Agent-based Modelling* (ABM) is connected to many fields and applications including complex systems science, computer science, and traditional modelling. It draws its conceptual origin from Artificial Intelligence (AI) and *Complex Adaptive Systems (CAS)* research (Holland, 2006).There is no strict or universal definition for what actually constitutes an agent within an ABM. In some systems, any individually identifiable component is considered an agent (Macal & North, 2006). However, a far more common trend in literature such as found in specific simulations like CAS and AI systems (Russell & Norvig, 2003) is to define agents as identifiable discrete objects that are self-contained and autonomous, which are responsible for managing their own interactions, states and data.

ABMs are being applied to larger scale systems, modelling more complex phenomena as data become more readily available at higher levels of detail. One such large-scale phenomenon that has attracted significant interest in the last two decades in Australia is the threat of invasion of Old World Screwworm Fly, *Chrysomya bezziana* (herein referred to as OWSWF) to the mainland. The introduction of this insect pest has the potential to devastate the Australian livestock industries through primary impacts, such as the immediate loss of production, and secondary impacts on industries related to the livestock (McKelvie, Hamal, & Reynolds, 1993).

The decision support requirement was originally addressed with the development of the existing Bio-economic SWF invasion model (D. G. Mayer et al., 1994). Although this model has provided valuable insight for policy makers, it has several key short-comings.

- The underlying simulation mechanism for estimating the numbers of female OWSWF is based around an adaptation of the CLIMEX Growth modelling package (Suthurst, Maywald, & Kriticos, 2007). This system deterministically calculates the number of OWSWF in a location based upon the input data layers. Although this approach is computationally efficient, it fails to capture the underlying biological processes and fails to provide a rich platform for the sensitivity analysis of factors that contribute to the lifecycle of the insect.
- The existing system makes use of a cellular automata platform to represent the two dimensional spatial characteristics of the SWF's dispersal. Each cell in the grid represents a 20km by 20km area and it assumes that the zone represented within is homogenous in terms of the biological parameters associated with the location. This low resolution fails to capture the differences in the contributing biological input and potentially over-simplifies the simulation of the invasion.
- The model uses deterministic template based estimation for simulating the dispersal. This approach may not capture well the randomness of the flies' dispersal.

In order to meet the challenges surrounding the development of a modelling system with increased levels of detail and immense processing overhead coupled with applying dynamic stochastic sampling, this research outlines the development of a high performance agent-based simulation of the OWSWF's lifecycle and dispersal by exploiting NVidia's CUDA GPU platform and its data-parallel programming paradigm.

Graphics Processing Unit (GPU) technology has developed rapidly in the last decade, and is responsible for providing the raw computational power to perform the large quantities of floating-point operations required to produce complex three dimensional graphics in real time. A modern GPU is developed around the *many-core* processing paradigm, where large numbers of specialised processor cores are used to achieve a high throughput of operations in parallel (Kirk & Hwu, 2010). GPU technologies are expected to be applied to agent-based modelling across a number of fields to provide highly detailed simulations with intense computational requirements. This was achieved initially by making use of the functionality provided by graphics programming libraries such as Microsoft's Direct3D and the OpenGL, that were primarily designed for producing graphics processing software. The use of these libraries for other purposes is referred to as *General Purpose Graphics Processing Unit* (GPGPU) programming in (Goddeke, 2005), which shows how the OpenGL library can be used for data-parallel general purpose computation. The work of Lysenko & D'Souza (2008) provides a framework for creating models that are capable of simulating massive number of agents by applying the GPGPU approach. Perumalla and Aaby (2008) demonstrated the capabilities of the GPGPU technique by implementing a range of different simulation paradigms, including a call-based game-of-life simulation and a mood diffusion model. Their analysis showed that problem spaces which can be broken down into independent, a-synchronous processes lend themselves to GPU execution. Since its introduction, the CUDA platform has provided a highly suitable environment for ABMs. D'Souza *et al.*(2009), Richmond *et al.* (2009) and Strippgen and Nagel (2009) all showed how CUDA can be applied to produce highly detailed modelling environments, however, none of these works demonstrated modelling on a national scale.

The work in this article outlines an approach for applying NVidia's CUDA to the implementation of a national-scale agent-based model for simulating an OWSWF invasion of Australia. The use of CUDA in this context is quite novel, as there are few examples of national-scale systems that make use of the CUDA technology. The following sections include a review of the OWSWF biology, the lifecycle model used to simulate the insect's biology and an outline of the agent-based CUDA implementation. Lastly the GPU approach is compared with an equivalent single threaded *Central Processing Unit* (CPU) implementation to assess the performance improvements and the efficiency of the approach in terms of its use of the available processing power.

2. MODELLING OWSWF BIOLOGY

2.1 OWSWF Lifecycle

The biology of the Old World Screwworm Fly has received significant study and research by (Atzeni, Mayer, Spradbery, Anaman, & Butler, 1994) who developed a detailed lifecycle-based model for the biology of the OWSWF. This lifecycle model was developed as part of a bio-economic modelling project to validate the output from a computationally less complex growth index-based simulation. The index-based approach was used in the bio-economic model because the lifecycle model developed was too computational intensive for execution on desktop computers of the time. The index-based system was shown to provide an acceptable approximation.

The implementation of the lifecycle-based model by (Atzeni, et al., 1994) uses a cellular automata approach to capture the spatial details of the OWSWF population. The Australian mainland is represented by a grid of 30,992 cells, where each cell represents a portion of a 20 x 20 km (400Sq.km) area. This grid structure provides the spatial representation for the OWSWF population and the means of capturing the varying biological characteristics. Within each of these cells, the OWSWF population is broken down into thirty-six cohorts that cover the Egg, Larval, Pupal and Adult stages of the flies' lifecycle. Using a series of deterministic equations that calculate the survival and development, the populations transition from one cohort to the next. These calculations are carried out on a daily basis (i.e. the simulation cycles with the 'day' as the basic unit of time) using weekly averaged biological data including minimum and maximum temperatures, vegetation and soil moisture. This basic process drives the lifecycle-based model with additional calculations such as livestock strikes and economic estimations built on top of the populations within the cohorts.

The agent-based implementation adopted in this paper builds upon this approach, using the same lifecycle model consisting of thirty-six cohorts to represent the population structure. The cellular automata representation was selected amongst other spatial representations as it provides the most efficient structure for a simulation of this scale. Approaches such as farm/property-based point representations (Happold et al., 2010) and irregular polygons that represent homogenous areas based upon the combined input of underlying data layers were explored, however they failed to provide any additional benefit to spatial representation within a model of this scale, while increasing the complexity of the mechanics surrounding the spatial representation. For the agent-based implementation, a 1km x 1km high-resolution grid of cells was chosen to capture the two dimensional spatial aspect of the insect's invasion. This higher resolution was chosen as it represents the maximum resolution feasible based upon data requirements of the insect's lifecycle and the memory available on graphics cards at the time of implementing the memory structures outlined in Section 3.

The new approach has improved the deterministic, cohort-based model, by using individual-based, stochastic calculations to move the insects from one cohort to another. This approach would normally be too computationally intensive to implement, especially for the higher resolution grid; however, the capabilities of the GPU technologies make this feasible. The calculations that determine the survival and the development of the insects are taken from the original implementation; however, they are applied on an individual basis (i.e. individual insect) using stochastically sampled values.

2.2 OWSWF Dispersal

The dispersal of the insect is primarily influenced by the vegetation cover and the availability of wounds suitable for oviposition by females (D. Mayer, Atzeni, Swain, & Stuart, 1995). Recent studies of OWSWF dispersal activity in the Mesopotamia Valley show that despite the seasonal fluctuation in overall OWSWF populations, they always remain within well vegetated areas (Siddig et al., 2005). In essence, this results in the OWSWF being mainly dispersed along the Euphrates and Tigris Rivers. Through the analysis of data from capture and release experiments, Mayer et.al. (1993) found that the Cauchy distribution provided a suitable estimation of the dispersal distance of OWSWF. In their approach, for each insect a random direction is selected using a uniform distribution and a distance sampled from a Cauchy distribution. The median for the Cauchy distribution can be set as a simulation parameter or it is set dynamically based upon the ratio of egg laying female OWSWF (i.e. those searching for places to lay eggs) and the available wounds in the source grid square. In most situations it is observed that the (Cauchy Distribution) median dispersal distance is usually in the order of 1-2km (D. Mayer, et al., 1993); however, with the pressures of wounds availability as the environment factors this median can be up to 15km. This is based around the idea that there will be little dispersal within the OWSWF population when host wounds are plentiful in their existing location. In the model developed in (D. G. Mayer, et al., 1994), the median dispersal distance is selected dynamically based upon the ratio of gravid OWSWF to oviposition sites available in the source grid square, with a range from 2km to 15km when the ratio of gravid flies to oviposition sites is 4:1. To capture the effects of vegetation and ovipoisition site availability, Mayer, Atzeni, Swain, & Stuart (1995) implemented a re-distribution step that re-allocates flies to grids squares that have higher vegetation and wound availabilities. The result of this process is that the OWSWF population is pulled into areas where availability is higher and vegetation cover is more favourable.

Due to its high computational requirement, the stochastic approach outlined in the previous paragraphs was never implemented in either the bio-climatic model (D. G. Mayer, et al., 1994) or the lifecycle simulation (Atzeni, et al., 1994) to produce spread *templates* for each median dispersal distance. These distances are applied deterministically to specify the proportions of the dispersing insects in the surrounding cells, with the re-distribution step applied dynamically. With the use of the GPU technology, the agent-based implementation developed in this work applies the stochastic approach at the individual level using the higher resolution 1km x 1km grid. This represents a key improvement over the existing implementation as it provides an ideal stochastic mechanism that captures the possibility for random, yet unlikely, dispersal behaviour inherent in nature.

3. C++/CUDA Implementation

3.1 Overview

The implementation is developed using a base program written in the C++ programming language and NVidia's Compute Unified Device Architecture (NVIDIA, 2010). CUDA provides its own run-time library that is accessed using extensions to the C programming language and is compiled into the base C++ program using NVidia's C Compiler (NVCC), which is provided as part of the CUDA developer package. The model is organised as a standard heterogeneous program, where the host manages the memory and data requirements for the simulation and the device (in this case a GPU) executes the core simulation. The simulation logic outlined in Section 2 is implemented within a series of CUDA C *Kernel* functions which execute by creating a dynamically allocated data structure that stores the agent state information and a series of statically allocated data layers that contain the bio-climatic input data. The basic pattern of execution, illustrated in figure 1.0, involves a weekly cycle where input data is updated at the start of each week and the simulation's logic is processed daily until the start of the next week.

3.2 SIMULATION KERNELS

The simulation's logic is executed on a CUDA enabled GPU through a series of kernel functions that are responsible for simulating different sections of the lifecycle representing a single day within the simulation. The daily cycle consists of three phases: the application of survival rates to the individual flies, the dispersal of female flies, and the promotion of the insects from one cohort to the next according the development stage calculations. The separation of the daily cycle into these three stages, and into three different kernels is done so that processing across all threads is synchronised between each stage. As mentioned in section 2.1, the model uses a cellular grid where each individual cell represents a homogenous area of the Australian mainland. The data structure that represents this arrangement is an Australia-shaped, flattened, ragged array, where each element contains the agent data relevant to that grid square. Although logic executed at the individual (insect) level, CUDA threads process this logic at the grid-square level. That is, all agent logic within a given grid square is processed by a single thread. This approach was chosen in place of having a single thread for each agent due to the overhead of random number generation (RNG). To avoid random number sample-space collisions (i.e. multiple threads sampling the same random number in the RNG's sequence), a random number generator is created (using the standard CUDA Random number generator libraries) for each thread. This incurs memory overhead as the state of each RNG is saved in the global memory store, thus limiting the number of threads that can be efficiently created within the simulation.

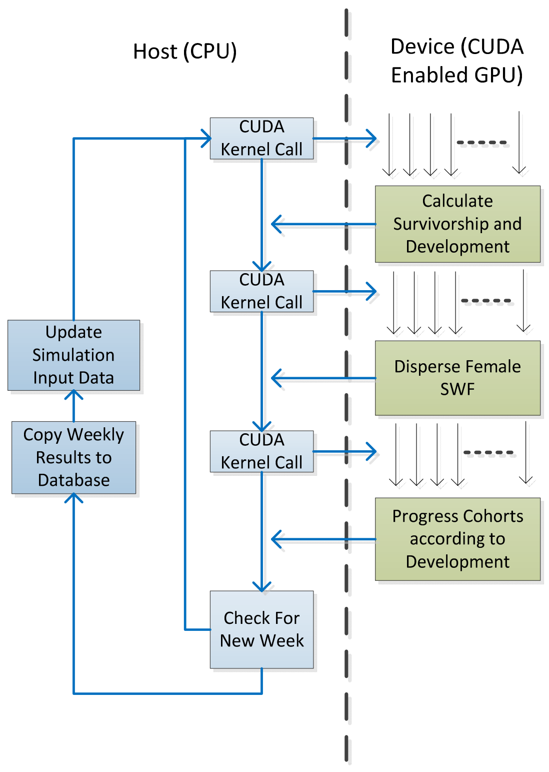
Host (CPU)
Device (CUDA Enabled GPU)
CUDA Kernel Call
Calculate Survivorship and Development
CUDA Kernel Call
Disperse Female SWF
CUDA Kernel Call
Progress Cohorts according to Development
Check For New Week
Update Simulation Input Data
Copy Weekly Results to Database

Figure 1.0 The Heterogeneous Program for the daily cycle structure within the simulation. The blue arrows represent the execution control by the host and the black arrows represent individual CUDA threads responsible for the model logic.

This structure also allows the number of threads used to process the set of grid squares to be independent of the number of grid squares. This is achieved by using each thread to process multiple grid squares in an iterative fashion. For example, the flat data structure that represents the Australian mainland will have 6,898,494 grid squares in this implementation. Instead of using an individual thread for each gird square (i.e.

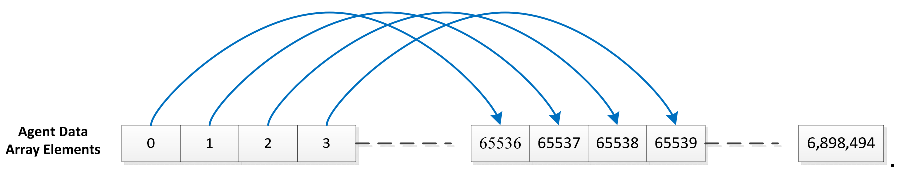

Figure 2.0 The execution of CUDA threads on multiple elements of a single dimensional array. Each thread has its own sample space within the array. Each thread starts at the index specified by its Thread Id and is incremented by the total number of threads to get the next element to process.

6,898,494 threads) a standard block of 65,536 (256 x 256) threads is used with each thread having its own sample space within the data structure of 6,898,494 items. Figure 2.0 illustrates this process. Within infested grid-square data-items, populations of up to approximately 2000 individual insect agents are present, spanning the entire SWF lifecycle.

4. PERFORMANCE ANALYSIS

Performance analysis of the simulation was carried out using a desktop PC running Windows 7 Ultimate 64-bit. The machine contained 16 GB of RAM, and an i7 processor with four physical cores. The database system used was PostgreSQL 9.1 server running on a 120 GB solid-state drive. The device used to execute the CUDA kernels (i.e. the model's logic) was an NVidia GeForce 680GTX with 2 GB of GDDR5 RAM. In order to analyse the effectiveness of the GPU implementation, an equivalent single threaded CPU implementation was produced from the logic of the GPU model. Average execution times were collected across multiple trial runs for completion of the simulation cycle processing. Figure 3 summarises the average execution times (in milliseconds) measured against the number of infested grid-square agents within the simulation.

From this experiment, it is evident that the CUDA implementation has achieved much lower execution times when compared to the CPU implementation as the number of agents within the simulation is increased. In addition to this, the CPU implementation shows a roughly proportional growth in execution time with the increase in the number of agents. This is expected, as processing is completely linear (i.e. the addition of agents will incur addition processing for each additional agent). In contrast to this, the CUDA implementation produces relatively static execution times across the range of agents measured in this experiment. This is indicative of the efficient hardware level scheduling of the threads processing.

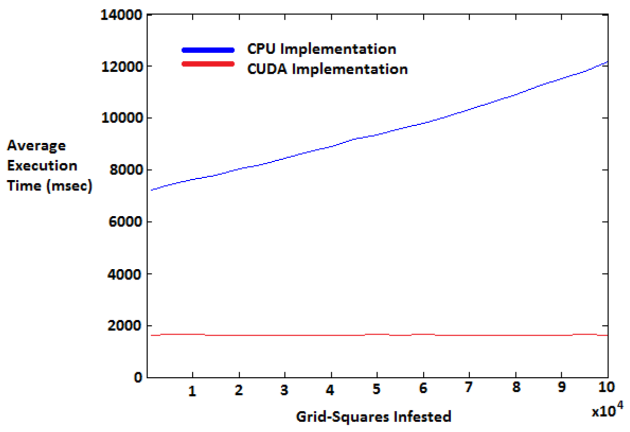

Figure 3.0 Comparison of the daily cycle execution time for different number of agents between the CPU and CUDA implementations.

5. CONCLUSIONS AND FUTURE WORK

The implementation presented in this work shows the potential for the application of the NVidia's CUDA GPU programming to the simulation of very large-scale complex systems. The CUDA platform has been successfully applied to produce an individual-based lifecycle model for the Old World Screwworm Fly within the Australian mainland, using 1km x 1km resolution. The system provides the ability to carry out detailed sensitivity analysis experiments on a range of lifecycle and bio-climatic inputs, with efficiency of execution that will allow for statistically significant number of trials. The approach de-couples the total number of agents from the number of CUDA threads used to process the logic within the model, providing the capability to run on a wide range of CUDA enabled devices. This scheme allows the model to make effective use of available resources such as available device main memory and supported compute capabilities of the device. The analysis and experimentation shows a substantial performance improvement when compared to a CPU-based implementation, achieving speed-up percentage that is almost proportional to the number of agents across the range of agents tested. The future work on this model will surround the implementation of the control and eradication measures and economic impacts of the invasion. These will be implemented within the existing CUDA kernels using a series of new agents within the simulation's logic. Additional analyses and development will continue to improve the efficiency of the data structures and the configuration of the execution.

REFERENCES

Atzeni, M., Mayer, D., Spradbery, J., Anaman, K., & Butler, D. (1994). Comparison of the predicted impact of a screwworm fly outbreak in Australia using a growth index model and a life-cycle model. *Medical and Veterinary Entomology, 8*(3), 281-291.

Berger, T. (2001). Agent-based spatial models applied to agriculture: a simulation tool for technology diffusion, resource use changes and policy analysis. *Agricultural Economics, 25*, 245-260.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America, 99*(Suppl 3), 7280-7287.

Bone, C., & Dragicevic, S. (2009). Evaluating Spatio-temporal Complexities of Forest Management: An Integrated Agent-Based Moddelling and GIS Approach. *Environmental Modeling and Assessment (2009), 14*(4), 481-496.

Connell, R., Dawson, P., & Skvortsov, A. (2009). Comparison of an Agent-based Model of Disease Propagation with the Generalised SIR Epidemic Model. *Air Operations Division: DSTO Defence Science and Technology Organisation, Australia*.

D'Souza, R., Marino, S., & Kirschner, D. (2009, March,2009). *Data Parallel Algorithms for Agent-Based Model Simulation of Tuberculosis On Graphics Processing Units.* Paper presented at the Agent-Directed Simulation Symposium.

Goddeke, D. (2005). GPGPU - Basic Math Tutorial. from http://www.mathematik.uni-dortmund.de/~goeddeke

Happold, J., Garner, G., Miron, D., Sajeev, A. S. M., Kwan, P., & Welch, M. (2010). Towards a national livestock disease model. In F. F. Australian Dept. of Agriculture (Ed.).

Holland, J. (2006). Studying Complex Adaptive Systems. *Journal of Systems Science and Complexity, 19*(1), 1-8.

Kirk, D. B., & Hwu, W.-m. W. (2010). *Programming Massively Parallel Processors* New York: Morgan Kaufmann.

Lysenko, M., & D'Souza, R. M. (2008). A Framework fo Megascale Agent Based Meld Simulations on Graphics Processing Units. *Journal Artificial Societies and Social Simulation, 11*(4 10).

Macal, C. M., & North, M. J. (2006). *Tutorial on agent-based modeling and simulation part 2: how to model with agents*. Paper presented at the Proceedings of the 38th conference on Winter simulation.

Mayer, D., Atzeni, M., & Butler, D. (1993). Spatial dispersal of exotic pests—the importance of extreme values. *Agricultural Systems, 43*(2), 133-144.

Mayer, D., Atzeni, M., Swain, A., & Stuart, M. (1995). Models for the spatial dispersal of insect pests. *Environmetrics, 6*(5), 497-503.

Mayer, D. G., Atzeni, M. G., Butler, D. G., Anaman, K. A., Glanville, R. J., Stuart, M. A., et al. (1994). Biological Simulation of a Screwworm Fly Invasion of Australia. *Project Report Series Q094005*.

McKelvie, L., Hamal, K., & Reynolds, R. (1993). Producer and consumer welfare effects of an invasion of screwworm fly in the Australian livestock sector. *ABARE report to the Queensland Department of Primary Industries*.

NVIDIA. (2010). *NVIDIA CUDA: Programming Guide*. Santa Clara: NVIDIA Corperation.

Perumalla, K. S., & Aaby, B. G. (2008). *Data Parallel Execution Challeneges and Runtime Performance of Agent Simulations on GPUs*. Paper presented at the Proceedings of the Spring Simulation Multi-Conference.

Richmond, P., Coakley, S., & Romano, D. (2009). *Cellular Level Agent Based Modelling on the Graphics Processing Unit*. Paper presented at the High Performance Computational Systems Biology HIBI'09. International Workshop on.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. New Jersy: Prentice Hall.

Schelhorn, T., O'Sullivan, D., Haklay, M., & Thurstain-Goodwin, M. (1999). *STREETS: an agent based pedestrian model*. Paper presented at the Computers in Urban Planning and Urban Management.

Siddig, A., Al Jowary, S., Al Izzi, M., Hopkins, J., Hall, M., & Slingenbergh, J. (2005). Seasonality of Old World screwworm myiasis in the Mesopotamia valley in Iraq. *Medical and Veterinary Entomology, 19*(2), 140-150.

Strippgen, D., & Nagel, K. (2009). *Usuing Common Graphics Hardware for Multi-Agent Traffic Simulation with CUDA*. Paper presented at the SIMUTools.

Suthurst, R. W., Maywald, G. F., & Kriticos, D. (2007). *CLIMEX Version 3: User Guide*. Melbourne: Scientific Software.