The Implementation and Visualisation of a Large Spatial Individual-Based Model

Tim Hopkins and David R. Morse
University of Kent at Canterbury
Computing Laboratory, Canterbury, Kent CT2 7NF, UK
Tel: +44 1227 823793, Fax: +44 1227 762811
e-mail: T.R.Hopkins@ukc.ac.uk D.R.Morse@ukc.ac.uk

Abstract

We look in detail at the implementation of a simulation of the spread of Barley Yellow Dwarf Virus in a barley field. The model considers explicitly each individual plant and aphid, therefore it requires special care to reduce the amount of storage used whilst still producing a computationally efficient code. We attempt to quantify the cost of some of the decisions made in terms of their memory and processor time requirements. Finally we briefly consider the visualisation of the results and how the amount of data produced by the model may be reduced to a manageable level.

Keywords

Barley Yellow Dwarf Virus, Simulation, Fortran 90

1 INTRODUCTION

The increased power and availability of computers has led to the development of two new types of ecological simulation model: individual-based models and spatially-explicit models. In the former, each biological entity in the model is simulated at the level of the organism rather than the population (see DeAngelis & Gross (1992), Judson (1994) and Kawata & Toquenaga (1994) for reviews). In the latter type of model, space is represented explicitly, with communication (or migration) often being restricted to between adjacent patches rather than global communication with all the patches in the model (see for example, Hassell, Comins & May (1991), Comins, Hassell & May (1992)). Both types of model use substantial computing resources, both processing power and memory (Villa 1992). Simulations which combine the two models tend to use parallel computers such as transputer networks (Morse 1993), or the Connection Machine (Costanza & Maxwell 1991). There have been few

Environmental Software Systems Vol.2 R. Denzer. D.A. Swayne & G. Schimak (Eds.)

examples of models which have been developed on general purpose computer hardware.

In the current paper we describe the implementation of an individual-based, spatially-explicit model for the spread of Barley Yellow Dwarf Virus (BYDV) (Power 1996). Because of the large amounts of data involved a naive approach is doomed to failure due to storage considerations. We consider a number of ways in which we can reduce the amount of memory required to store data and discuss some of the implications these may have on the resultant run times. Finally, we consider how the results of the simulation may be visualised. A more detailed version of this paper may be found in Hopkins & Morse (1996).

2 DETAILS OF THE SIMULATION

Barley Yellow Dwarf Virus is transmitted between plants by aphids such as *Rhopalsiphum padi* (L.) feeding on the phloem of the host plant (Power 1996). Morgan (1989) gives two ways in which the disease is spread by the aphids. By primary infection which occurs when infective, winged (alate) aphids migrate onto the crop from reservoir populations of the virus; or by secondary infection resulting from dispersal of the offspring of the migrant aphids. Note that these migrant aphids first have to acquire infection by feeding on an infected plant before they can pass it on to another plant.

We attempt to simulate the spread of BYDV in a cereal field by keeping track of the position and state of the individual aphids and by considering their effect on individual barley plants. The simulation consists of a sequence of days during which the following events take place in the order indicated

1. Immigration of aphids: over a defined period of consecutive days at the beginning of the simulation period a predetermined number of winged aphids, all of the same age, are randomly placed on individual barley plants in the field. A probability threshold is provided which determines which of the immigrant aphids are infected with BYDV.
2. Based on given daily temperature data, the development and reproductive rates for all the aphids in the simulation are calculated.
3. For each aphid in the field,
   (a) Its age is updated based on the daily development rate calculated in step 2 above. Newly born aphids become wingless morphs which die when their computed 'ages' exceed one.
   Note: no winged aphids develop during the simulation.
   (b) Based on the daily reproductive rate, newly born aphids appear on the same plant as their parents.
   Note: newly born aphids immediately become part of the population: their ages are updated and they may move plants.
   (c) Its position may change; movement occurs when a given probability threshold is exceeded. The current simulation allows a choice of two

dispersal models: *purely random*, where the aphid is transported to a random point in the field; and *restricted movement*, where movement is restricted to a nearest neighbour move. Probabilities of 0.1 for the four diagonal plants and the two in the North and South directions, and 0.2 for the two plants in the East and West directions were used. These probabilities were chosen as a first approximation in modelling the tendency of aphids to move between plants in the same row in a cereal field (Power 1996). This movement preference reflects the fact that inter-plant spacing is closer within rows than it is between rows of plants.

4. The virus states of the aphid and plant on which the aphid is feeding are updated. An infected aphid always feeds upon its current plant, passing on the infection. A healthy aphid is always infected if it settles on an infected plant. The incubation periods for both plant and aphid are given and have been assumed to be constant in time.
5. We are primarily interested in visualising the spread of the virus through the cereal field and the population dynamics of the aphids (the position and number of infected aphids and the general population are also of interest).

3 STORAGE REQUIREMENTS OF THE MODEL

In order to provide a realistic model we require the cereal field to contain between one and ten million plants. This corresponds to a planting density of 300 plants/m^2 and a field plot which is of the order of $100m^2$. For 10^6 plants we specified an immigrant population of 2000 aphids/day for four days and for 10^7 plants, 20000 aphids/day for four days. The resultant populations after 30 days were 8×10^5 and 8×10^6 aphids respectively.

In the simulation model we store, for each aphid, its current age (a real value in the range [0,1]), its life stage (one of newly born, wingless, winged or dead), its position in the field (x, y coordinate – a pair of integers), its BYDV status (one of infected, incubating, or uninfected) and, finally, its incubation period (the number of days the virus has been incubating – this information is used to update the BYDV status from incubating to infected).

For each plant, we record its BYDV status (one of infected, incubating, or uninfected) and its incubation period (the number of days before a plant, bitten by an infected aphid, itself becomes infected).

A naive storage scheme would require an extremely large amount of memory. However, the data within the aphid record can be compressed, decreasing the storage required, at the cost of packing and unpacking the data. For example, the aphid data could be represented as follows, where the last four fields may be packed into a 32 bit integer.

- age = 32 bits (real)
- life status = 2 bits (only 4 possible states)

- position in the field = 24 bits (range $0 \rightarrow 10^7$ i.e., $\approx 4000 \times 4000$ plants)
- BYDV status = 2 bits (only 3 possible states)
- incubation period = 4 bits (allows up to 15 days).

We note that although the age of the aphid requires updating at each time step, if no other data associated with that aphid changes, there is no need to repack the compressed data.

To avoid the need to reserve array space in advance, we implemented the storage of the aphid data as a linked list of records, where each aphid has its age, life state, position, BYDV status and incubation period recorded. The major disadvantage of a simple linked list over an array is the loss of fast random access to the data. However, in this application, we are only interested in accessing the aphids sequentially. There are also time overheads involved in packing and unpacking the records to extract the required data, and the hidden storage overhead of the pointers involved in the linked list.

We require to access the plant data randomly (typically we wish to ascertain efficiently whether the plant at a given coordinate is infected) so linked lists cannot be considered practical because of their sequential access mechanism. Provided we temporarily store the positions of the incubating plants in an efficient manner we may reduce the plant array to a bit array. If the incubation period is short (4 days is typical), and not too many plants are incubating at any time, we may also store the incubating plants as linked lists. This has a storage overhead of an integer and one pointer for each incubating plant.

The final structure used for storing the plant data was a bit array for the infected/uninfected state of each plant along with a circular list of linked lists storing the coordinates of incubating plants. On completing the incubation period these plant lists are used to update the bit array and the storage is reused. Use of a circular list at the head of the data structure allows easy updating of the pointers recording the current day and end of incubation day pointers.

An alternative method of storing the plant data would be to allocate an extra incubation bit for each plant. This method is only more efficient if more than 1.5% of all the plants are infected over any incubation period.

4 PERFORMANCE AND VISUALIZATION

Fortran 90 (ISO/IEC 1991) was chosen as the implementation language as it offered portability, pointers, bit manipulation and information hiding along with the possibility of porting the code on to a parallel architecure by using HPF (Koelbel, Loveman, Schreiber, Steel Jr. & Zosel 1994).

All code was developed on a Silicon Graphics Indy with a 133MHz R4000 processor and 32Mbytes of memory running Irix 5.3 and using the Edinburgh Portable Compiler Fortan 90 system. It was also successfully compiled and run on a Sun Sparc 10 using the Craysoft and NAG Fortran 90 systems without any source code changes.

The results reported in this section refer to simulations on square grids of p plants in each direction (i.e., p^2 plants in total) with $0.002p^2$ immigrant aphids on each of the first four days (days 70–73). The immigrant aphids are all winged aphids aged 0.5 with a probability of 0.1 of being infected with BYDV. The probability that an aphid moves during the course of a day is 0.05. The simulation takes place between days 70 and 100.

Figure 1 shows the total CPU time used and the final number of aphids for $p = 500$, 1000 and 2000 using a linked list to store the aphid information and packing the coordinate, life stage, incubation time and BYDV status into a single integer.

# Plants	# Immigrant Aphids	Final # Aphids	CPU seconds
2.5×10^5	2000	2×10^5	8.5
10^6	8000	8×10^5	41
2.5×10^6	20000	1.8×10^6	130
4×10^6	32000	3.2×10^6	460

Figure 1 Results on 133MHz SG Indy using EPC Fortran 90 with highest level of optimization

A more detailed breakdown of these timings shows that as p increases, the CPU time required to process each day of the simulation increases far more rapidly towards the end of simulation. The reason for these large jumps at day 99 for $p = 1500$ and at day 96 for $p = 2000$ may be explained by the dramatic increase in the number of page faults (from 1 when $p = 1000$ to 96,800 when $p = 2000$) which take place as the data structures grow. It may also be the case that the run time system is attempting to free extra space by garbage collecting. In real terms, for $p = 2000$, this book-keeping work by the system is accounting for around 50% of the processor time used for the simulation.

One area where data storage could be improved is when a plant is bitten by more than one aphid. This results in multiple copies of the plant's coordinates being stored in the linked list. This problem becomes more pronounced as the aphid population grows but the excess is quite small compared to the storage required for the extra bit per plant necessary to prevent duplicate entries.

Packing the aphid records into an integer and a real as described in Section 3 did allow larger problems to be tackled. Tests indicated that the overhead associated with packing is approximately 15%.

Finally the simulation was implemented using an array of packed records; this provided a saving in space but required setting the length of the aphid data array at the start of the simulation. The execution times are comparable for smaller values of p but for the larger domains ($p = 1500$ and $p = 2000$) they do not exhibit the page fault problem to the same degree (just 10,000 page faults for $p = 2000$). The run time for $p = 2000$ was 200 seconds with the effects of swapping not evident until day 97. The delayed effects of swapping

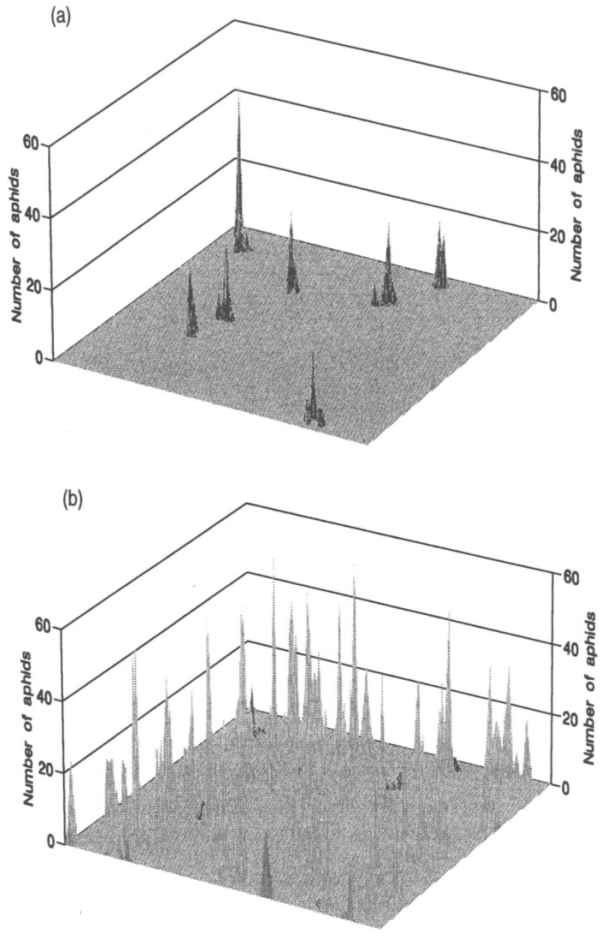

Figure 2 (a) Healthy and (b) infected aphid numbers on each plant in a simulation on a 100 × 100 lattice. The dark grey areas denote plants which have been infected by the BYDV virus. (The size of the simulation was reduced for clarity on a small figure.)

are due to the fact that using arrays does not incur storage overheads for pointers.

One view of the output of the simulation is an animation of the spread of infected plants and aphids as the simulation progresses. Overlaid on top of this could be contour maps or iso-surfaces showing the distribution and abundance of the aphids — both the infected and the uninfected populations.

A single frame from such an animation (corresponding to the state of the simulation at day 100) is shown in Figure 2.

Generally, state information stored in arrays (such as the plant infection status) could be extracted simply by printing the arrays at each time step. Other information was more difficult to extract, such as the aphid data (which was stored in linked lists) although summary information could be accumulated at each time step as each aphid was processed. Relating the aphid and plant states and the changes in state and movement of the aphids was particularly difficult and involved generating extremely large data files which were then post-processed by simple programs to extract the relevant data in a form which could be visualized.

In general it proved simpler to separate physically the simulation and visualization functions of the modelling process by using graphical and visualization packages such as Uniras and Explorer. These were linked to the simulation using temporary files. An alternative approach is to include bespoke graphical display facilities in the simulation. While this would have made the simulation easier to use, it reduces the potential for analysing the simulation output in non-standard ways.

5 SUMMARY AND CONCLUSIONS

We have used Fortran 90 to produce an efficient implementation of a model for the spread of BYDV within a cereal field. Pointers and bit level intrinsic functions allowed us to use data structures that grew with the aphid population and to compress the data required to record the state of the aphids and plants. This approach led to CPU time overheads due to page faults and the packing and unpacking of data. The use of arrays of records reduced the number of page faults.

One of the problems with individual-based models is that of visualizing the vast amounts of data that describe the state of the model at each stage of the simulation. With our model we were primarily concerned with the spread of the disease within the plants. We could thus just use the coordinates of any newly infected plants to update our view of the plants at the end of each time step. To obtain a detailed view of the aphid population is more complex. Some information like the number of aphids at each life stage or the number of infected and uninfected aphids can easily be updated as the aphid data is sequentially processed. Other views of the data, especially those requiring links between the plants and the aphids were more difficult to obtain and generated very large data files. Improvements may be possible in this area by recording aphid movements and state changes, and post-processing these to form animations.

Future work in this area will investigate a distributed code using HPF, an object oriented approach and a more realistic model. We will also be considering how we can couple the computation and visualization parts of the simulation more tightly together. Such a coupling will allow a more interac-

tive exploration of the behaviour of the simulation under various parameter combinations and changes rather than a post-hoc analysis of the results.

REFERENCES

Comins, H., Hassell, M. & May, R. (1992), 'The spatial dynamics of host parasitoid systems', *Journal of Animal Ecology* **61**(3), 735–748.

Costanza, R. & Maxwell, T. (1991), 'Spatial ecosystem modeling using parallel processors', *Ecological Modelling* **58**(1–4), 159–183.

DeAngelis, D. & Gross, L. (1992), *Individual-based models and approaches in ecology*, Chapman & Hall, London.

Hassell, M., Comins, H. & May, R. (1991), 'Spatial structure and chaos in insect population-dynamics', *Nature* **353**(6341), 255–258.

Hopkins, T. R. & Morse, D. R. (1996), The implementation and visualisation of a large spatial individual-based model using Fortran 90, Technical Report 18-96, University of Kent, Canterbury, UK.

ISO/IEC (1991), *Information Technology – Programming Languages – Fortran (ISO/IEC 1539:1991(E))*, ISO/IEC Copyright Office, Geneva.

Judson, O. (1994), 'The rise of the individual-based model in ecology', *Trends in Ecology & Evolution* **9**(1), 9–14.

Kawata, M. & Toquenaga, Y. (1994), 'From artificial individuals to global patterns', *Trends in Ecology & Evolution* **9**(11), 417–421.

Koelbel, C. H., Loveman, D. B., Schreiber, R. S., Steel Jr., G. L. & Zosel, M. E. (1994), *The High Performance Fortran Handbook*, MIT Press, Cambridge, Mass.

Morgan, D. (1989), 'A simulation model of BYDV epidemiology', *Proceedings of CYMMIT Workshop on BYDV — 1987* pp. 300–304.

Morse, D. (1993), Spatial simulation modelling of insect population dynamics on a transputer network, *in* J. Kerridge, ed., 'Transputers and occam research: new directions', IOS Press, Netherlands, pp. 66–75.

Power, A. (1996), 'Competition between viruses in a complex plant-pathogen system', *Ecology* **77**(4), 1004–1010.

Villa, F. (1992), 'New computer architectures as tools for ecological thought', *Trends in Ecology & Evolution* **7**(6), 179–183.

6 BIOGRAPHY

Tim Hopkins is a Reader in Numerical Computation and is Algorithms Editor for ACM Transactions on Mathematical Software. He has a Ph.D. from the University of Liverpool. His main research interests are in parallel numerical computation and software code quality metrics.

David Morse is a Lecturer in Computer Science and is secretary to the British Ecological Society's Ecological Computing Group. He has a Ph.D. from the University of York. His main research interests are in parallel and distributed simulation modelling and mobile computing.