Simulated Evolutionary Stable Strategies

James Hammerton

EPCC–SS94–08

Simulated Evolutionary Stable Strategies

James Hammerton

EPCC–SS94–08

September 1994

Abstract

The aim of this project was to develop an individual based predator-prey simulation, where each individual animal could follow a different strategy, and where the strategies would compete. For simulations involving thousands of individuals or more, this requires parallel processing. The problems to solve included how to obtain information about other individuals in the same field and how to obtain information about neighbouring fields, without incurring large communications overheads. Solutions to these problems were found, and several experiments run with the simulator, involving competing migration strategies. Most of the time having all the animals involved follow a selfish strategy seemed to be stable — but on one occasion a mixture of two strategies seemed to be stable. This project opens up the possibility of further work in this area, with more realistic simulations involving animals with independent evolveable strategies.

1 Introduction

When trying to explain the behaviour of animals, Darwin's survival of the fittest theory has been very successful. However there's always been the ambiguity of whether you explain the behaviour of the animal in terms of its interests or the the interest of the species or group. However recently, Richard Dawkins proposed the "Selfish Gene" theory [Dawkins, 1978] which attempts to explain the behaviour in terms of what's best for the genes an animal carries. In this view, an animal is a carrier for the genes and the genes influence its behaviour.

An interesting idea that arises from this is that where it might be in the animal's best interests to act selfishly, it may be in the genes' best interests to have a degree of altruism, since the other members of the species would be carrying the genes (or a portion of them) as well. This may be because if more than a certain portion of the population of a species behaves selfishly then collectively the species, and therefore the genes being carried by the species, are threatened. Thus whilst it may be that the best interests of an animal would require one particular behavioural strategy, the best interests of the genes might require a division of time between strategies, or a division of labour between strategies. This could give rise to the evolution of Evolutionary Stable Strategies (ESSs) in a population. An ESS is defined as a strategy or set of strategies, which persist in a population, with a certain distribution, deviation from which would be penalized by natural selection. The idea is that it would be in the genes' best interests if say 30% of the population followed one strategy, 40% another and 30% another one still. ESSs were first proposed for explaining animal behaviour by John Maynard-Smith and G.R. Price[Maynard-Smith and Price, 1973], and Richard Dawkins has explained how the selfish gene and ESSs can be combined to explain animal behaviour.

The aim of this project is to build a simple predator-prey simulation which allows various strategies to compete and to see if any ESSs occur, and thus allow a direct testing of the ESS theories. Standard simulation models however are analytic and deterministic. They rarely include the behaviour of each animal in their models. This also makes it difficult to simulate the effects of having charateristics unique to specific individuals. In contrast, a new approach is being developed which does allow these characteristics to be simulated. This approach involves using the power of computing, particularly parallel computing, to simulate the interactions between thousands of individuals stochastically.

There has already been some success in running spatially based simulations on parallel machines[Smith, 1991]. These simulations have a data structure consisting of a grid of "fields" each with associated numbers of predators and prey which can interact with each other, or migrate to other fields. It is these "fields" which are spread over the processors on a parallel machine, rather than the individual animals themselves. Thus each processor will be handling a field with an associated number of predators and prey. Communications between processors will only occur when migrations occur. These spatial models however can suffer from poor load balancing, and whilst they do model interactions between individuals, it is still difficult to track specific individuals and therefore to model genes as they evolve. For this reason, work has been done on creating individual based models, where the processors handle a collection of individual animals, each with their own characteristics, and some success has been achieved with these already[Smith, 1993] in terms of seeing how a single strategy fares in a population which has "dumb" animals which move randomly and "intelligent" animals which follow the strategy. This project's aim is to extend the individually based model to handle competing strategies and in the long run, evolveable parameters for each individual.

Section 2 gives details of the preliminary work for this project, section 3 discusses how competing strategies were added to the basic model, section 4 describes the experiments done with the model, section 5 gives the project's conclusions and section 6 outlines some possible areas for further work.

2 Preliminary Work

This section details the preliminary work that was required to provide a predator-prey model, implemented as a spatially based program which the later programs were compared against, and the reimplementation of the model as an individually based program. These programs and all the experiments in this project, were run on the Edinburgh Parallel Computing Centre's Connection Machine, using Connection Machine Fortran.

2.1 Implementing a spatially based simulation

In order to provide a program which could be used as a benchmark for the individual based simulation, a spatially based simulation was created which included births, deaths, migrations and predations. The program was written in a data parallel manner using Connection Machine Fortran. The data structure is a 2d array of fields, each with an associated number of predators and prey. The basic algorithm for this appears below:

```
Main loop
---------
  read in parameters.
  distribute the populations
  for i = 1 to number of iterations do
    do births and deaths
    do predations
    do migrations
  end

Births and deaths
-----------------
  Pbirth = probability of birth
  Pdeath = probability of death
  for each field do in parallel
    for each animal do
      random = random number between 0 and 1
      if (random < Pbirth) then
        give birth
      else if (random < 1-Pdeath) then
        die
      end if
    end
  end

Predations
----------
  Ppred = probability of predation
  set all predators to hungry
  for each field do in parallel
    for each predator do
      random = random number between 0 and 1
      if (random < Ppred and number of prey > 0) then
        set predator to satisfied
        kill a prey
      end if
    end
  end

Migrations
----------
  Pmigrate = probability of migrating
```

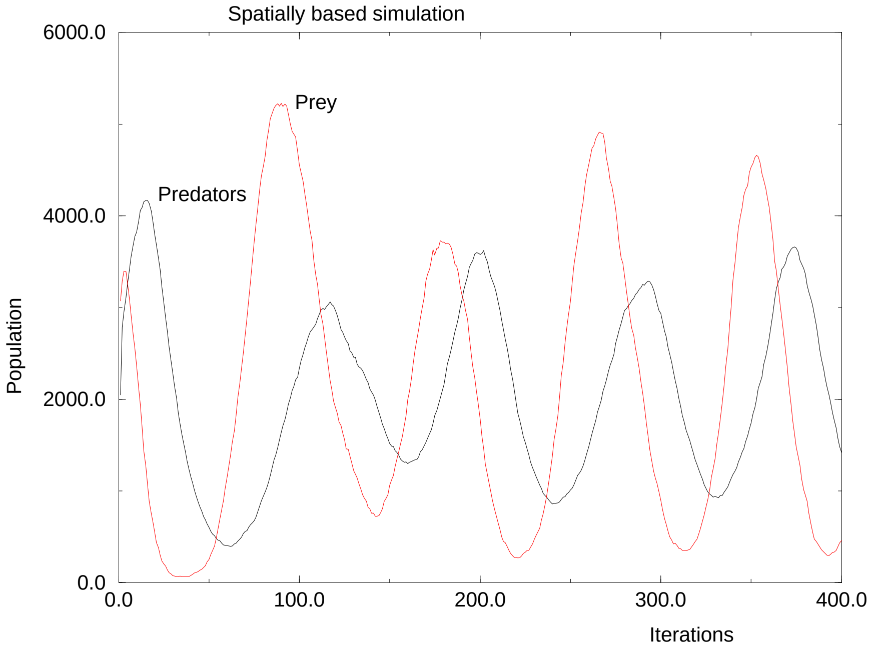


Figure 1: Graph showing how population levels of predators and prey vary over time in the spatially based simulation.

```
for each field do in parallel
  for each animal do
    random = random number between 0 and 1
    if (random < Pmigrate) then
      choose a random direction(N, E, S, or W)
      add 1 to animals moving in that direction
    endif
  end
end

use CM Fortran's cshifts to perform the migration
```

The Connection Machine(CM) Fortran's whole array operations were used for parallel operations. Hungry predators would be more likely to die than satisfied predators, hence failing to catch a prey would incur a penalty. The code is simple, and captures the essence of what is going on in a predator-prey situation — the two populations can interact, migrate, give birth or die. A graph of the variations in the population levels can be seen in Figure 1.

Note the oscillations of the population levels and that they oscillations for predators are roughly 90 degrees out of phase with those for prey. This is what the standard Lotka-Volterra model[Lotka, 1925; Volterra, 1926] predicts and is suggestive that the program has captured the essential components of the predator prey simulation. Reproducing these oscillations will be essential for verifying the validity of the individual based model.

2.2 Implementing an intial individual based model

The next stage was to reimplement the above model but in an individual based manner. Looking at the spatially based code code above, should bring out several important features:

1. When a successful predation occurs, the fact that an animal has just been killed is available for use in further predations to ensure that no animal is "killed" twice.

Entry	Usage
type	Determines whether the animal is a predator or prey
hungry	Whether the animal is hungry or not
Xloc	X coordinate of the animal's grid location
Yloc	Y coordinate of the animal's grid location
survivors	Whether the animal is dead, alive or has given birth
total	Array index to send the animal to after an event
newcells	Indicates where new fields start in the arrays
newspecies	Indicates both where new fields start and where the type of animal changes

Table 1: Core data structure for individual based simulation

2. The data structure is static, despite the fact that the number of animals can grow and shrink.
3. Information about the individuals in a field is available locally to the processor on which that field resides, and therefore to the other individuals in the field.

An individual based simulation will have a data structure consisting of a set of animals spread over the processors, as opposed to a set of fields spread over the processors. As a result of this, 1 and 3 no longer apply without involving the transfer of information between processors. Also since the total number of individuals will expand and contract, the data structure will be dynamic and therefore 2 will no longer apply. This therefore gives us the following problems to solve:

1. We need an efficient way for each individual to obtain information about the other individuals in the same field.
2. We need an efficient way of performing predations — actually pairing off each predator and prey will involve large communications overheads.
3. We need to have a dynamic data structure that can expand and contract as the populations grow and decline.

2.2.1 The basic algorithm

Problems (1) and (3) were tackled by Mark Smith[Smith, 1993], and a variant of the algorithm he used forms the basis of this program. His solution was to have a data structure where each individual was represented by a record with several entries, and the totality of the individuals represented as an array of such records. The entries are held locally on the same processor, and thus the aim is to ensure that as far as possible, all the information required for a particular phase of the program is stored in the entries. This gives rise to the data structure depicted in Table 2.2.1. This data structure is the core of that used in the later parts of this project.

The basic algorithm follows:

```
read in parameters
sort array by X location, Y location and type of animal
for i=1 to number of iterations
  do births and deaths
  reconstitute arrays
  do predations
  reconstitute arrays
  do migrations
  resort arrays
end
```

Births and deaths were performed simply by setting the number of survivors to 0(for the animal dying), 1(for no change) and 2(for giving birth). Migrations were performed by choosing an axis to move along(ie

X or Y), and then adding or subtracting 1 from that axis. The reconstitution of the arrays and the sorting of the arrays take this information into account to create the new individuals, get rid of the animals that have been killed and ensure that individuals in the same field form a contiguous block within the arrays. This leaves the reconstitution of the arrays and predations. Predations are explained in Section 2.2.3. Reconstitution of the arrays is explained Section 2.2.2.

2.2.2 Reconstituting the arrays

We need to ensure that the information an individual holds about the other individuals in its field is consistent with the information the others hold. To ease the communications, we therefore also want to ensure that the individuals are close together.

- Counting up the number of predators and prey(from now on foxes and rabbits, as used in the program) in a field.
- Copying the number of predators/prey to all individuals concerned.
- Making space for individuals that give birth.
- Allowing the data structure to contract when deaths have occurred.

The solution Mark Smith found, was to use the "total" field of the data structure to create a running total of the "survivors" field. This was then used as the array index to send the individuals to. Where survivors was 0, the individual was ignored, since it was dead, where it was 1 or 2, it was sent to the index in the corresponding "total" field. After this was performed, the individuals with the survivors element set to 2 were copied to the adjacent index in the array. An example is given in Figure 2. Next, the number of foxes and rabbits in each field are counted up, and the information copied to all the individuals in that field. This can be done efficiently on the Connection Machine, using its in-built add and copy scans with the newcells and newspecies fields as masks for the scans. Figure 3 should make this clear. Further details of this can be found in [Smith, 1993]. Sorting of the array after migrations was done using the Connection Machine's built in sorting functions.

2.2.3 Performing the predations

The problem with pairing off a predator and prey to perform a predation is that a predation changes the number of prey in a field, and if we are to avoid having a prey being killed twice, we need to convey this information to all the other individuals, and these individuals will probably reside on other processors. However if we simply decide that a predator has a probability of being satisfied, and that a prey has a probability of being killed, we can avoid this. We do however want to ensure that the expected number of predators satisfied and prey killed are equal and that these are equal to the expected number of predations. Below is a proof of a method for doing this:

The expected number of predations(assuming each fox gets only one chance at catching a rabbit, and each rabbit only gets one chance to be caught by a fox) is:

$$Ep = min(Pred, Prey) \times P(predation)$$

where $P(predation)$ is the probability of predation, $Pred$ is the number of predators in the field, and $Prey$ is the number of prey in the field.

If we let the probability of a prey being killed be:

$$P(preykilled) = \frac{P(predation) \times min(Pred, Prey)}{Prey}$$

then the expected number of prey caught will be:

$$Eprey = P(preykilled) \times Prey$$

index	1	2	3	4	5	6	7	8	9
type	F	F	R	R	F	R	R	-	-
Xloc	1	1	1	1	1	1	1	-	-
Yloc	1	1	1	1	2	2	2	-	-
Survivors	1	2	1	0	2	1	2	-	-
Total	-	-	-	-	-	-	-	-	-

Stage 1: The setup after births and deaths have been performed.

index	1	2	3	4	5	6	7	8	9
type	F	F	R	R	F	R	R	-	-
Xloc	1	1	1	1	1	1	1	-	-
Yloc	1	1	1	1	2	2	2	-	-
Survivors	1	2	1	0	2	1	2	-	-
Total	1	3	4	4	6	7	9	-	-

Stage 2: Calculating the array index to send each element to.

index	1	2	3	4	5	6	7	8	9
type	F	-	F	R	-	F	R	-	R
Xloc	1	-	1	1	-	1	1	-	1
Yloc	1	-	1	1	-	2	2	-	2
Survivors	1	-	2	1	-	2	1	-	2
Total	1	-	3	4	-	6	7	-	9

Stage 3: Sending the elements, where survivors $\neq 0$, to the index in total.

index	1	2	3	4	5	6	7	8	9
type	F	F	F	R	F	F	R	R	R
Xloc	1	1	1	1	1	1	1	1	1
Yloc	1	1	1	1	2	2	2	2	2
Survivors	1	2	2	1	2	2	1	2	2
Total	1	3	3	4	6	6	7	9	9

Stage 4: Copying all the elements where survivors = 2, into the adjacent empty positions.

Figure 2: Example of array reconstitution; F means Fox, and R means Rabbit. Afterwards, all the elements of the survivors array are set to 1, unless there is no animal there, and all the elements of total are set to 0.

$$
\begin{aligned}
&= P(predation) \times \frac{min(Pred, Prey)}{Prey} \times Prey \\
&= P(predation) \times min(Pred, Prey) \\
&= Ep
\end{aligned}
$$

Likewise if we let the probability of a predator being satisfied be:

$$P(predatorsatisfied) = \frac{P(predation) \times min(Pred, Prey)}{Pred}$$

then the expected number of predators satisfied is:

$$
\begin{aligned}
Epred &= P(predatorsatisfied) \times Pred \\
&= P(predation) \times \frac{min(Pred, Prey)}{Pred} \times Pred \\
&= P(predation) \times min(Pred, Prey) \\
&= Eprey \\
&= Ep
\end{aligned}
$$

index	1	2	3	4	5	6	7	8
type	F	F	R	R	F	F	R	R
Xloc	1	1	1	1	1	1	1	1
Yloc	1	1	1	1	2	2	2	3
Foxes	-	-	-	-	-	-	-	-
Rabbits	-	-	-	-	-	-	-	-
Newcells	T	F	F	F	T	F	F	T
Newspecies	T	F	T	F	T	F	T	T

Stage 1: The setup for counting number of foxes and rabbits.

index	1	2	3	4	5	6	7	8
type	F	F	R	R	F	F	R	R
Xloc	1	1	1	1	1	1	1	1
Yloc	1	1	1	1	2	2	2	3
Foxes	2	1	-	-	2	1	-	-
Rabbits	-	-	1	2	-	-	1	1
Newcells	T	F	F	F	T	F	F	T
Newspecies	T	F	T	F	T	F	T	T

Stage 2: After the add scans have been performed.

index	1	2	3	4	5	6	7	8
type	F	F	R	R	F	F	R	R
Xloc	1	1	1	1	1	1	1	1
Yloc	1	1	1	1	2	2	2	3
Foxes	2	2	2	2	2	2	2	0
Rabbits	2	2	2	2	1	1	1	1
Newcells	T	F	F	F	T	F	F	T
Newspecies	T	F	T	F	T	F	T	T

Stage 3: After the copy scans have been done.

Figure 3: Counting the number of foxes and rabbits in each field. Segment masks tell the Connection Machine where to restart the scans. In stage 2, Newspecies is used as the segment mask, in stage 3, Newcells is used.

The above gives rise to the following (parallel) pseudo-code, where “Predators” is an array containing the number of predators in the field each individual occupies, “Prey” holds the number of prey in the field, “Predation” is an array holding the probability of predation for each individual and “Min” is an array holding the minimum of the number of predators and number of prey in the field each individual occupies:

```
Min = min(Predators,Prey)
for all p that are predators
   r:= random number between 0 and 1.
   if (r < (Predation * min / Predators)) then
     p is no longer hungry

for all p that are prey
   r:= random number between 0 and 1.
   if (r < (Predation * min / Prey)) then
     kill p
```

The program implemented the same model(barring the different methods for predations) as the spatially based simulator, and Figure 4 is a graph of the program in action, given the same inputs as for Figure 1 for the spatially based simulation.

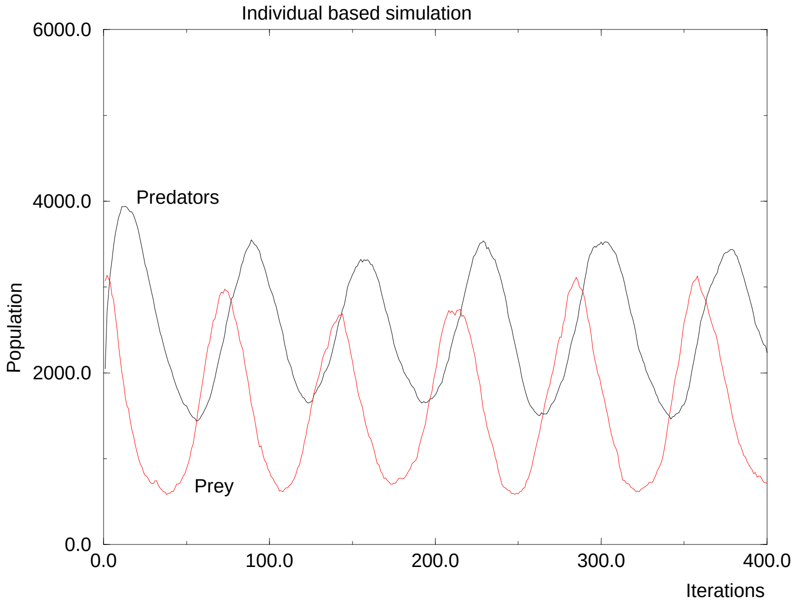


Figure 4: Graph showing how predator and prey populations vary over time on the individual based simulation

Note that the oscillations predicted by the Lotka Volterra model are again reproduced but that they are a lot more regular than the oscillations in the spatially based model. Also the predator population is generally higher than the prey population; in the spatial model the predator population is roughly the same as here, but the prey population oscillates more strongly and the predator population always falls within the range of the prey population's peaks and troughs.

2.2.4 Explaining the differences in behaviour

The fact that the individual based simulation produces the Lotka Volterra oscillations suggests that it has captured the essential components of the predator-prey situations, and is therefore a valid simulation. However the differences between the two graphs need to be explained.

The difference between the models used in the two simulations, is in the way the predations are done in them. Not only does the individual based model opt for simply making a certain number of predators become satisfied, and a certain number of prey get caught without necessarily matching them, but the probability of a prey being caught or a predator becoming satisfied also differs. This can be seen in Table 2. As the table shows, there is a greater likelihood that a rabbit will get caught in the spatially based simulation than in the individually based simulation, and this causes the stronger oscillations of the rabbit population. The reason for the differing probabilities, is that in the spatial simulation, if a rabbit is not caught by one fox, it might be caught by the next(assuming all are in the same field), whereas in the individually based simulation this is not the case. This has the effect of increasing the probability of a predation occurring. It also increases the expected number of predations over that given in the proof above.

2.3 Comparing the performance of the two simulations

The changeover to the individual based simulation, whilst likely to achieve better load balancing, making it easier to track individuals and give them unique characteristics, does involve a costs in increased

	Cases	
Rabbits	2	1
Foxes	1	2
Spatial simulation		
P(first rabbit caught)	P	P+(1-P)P
P(second rabbit caught)	0	-
P(first fox satisfied)	P	P
P(second fox satisfied)	-	(1-P)P
Individual simulation		
P(first rabbit caught)	P/2	P
P(second rabbit caught)	P/2	-
P(first fox satisfied)	P	P/2
P(second fox satisfied)	-	P/2

Table 2: Probabilities of a rabbit being caught, or a fox being satisfied in the two simulations. P is the predator power.

Grid Size	16x16	32x32	64x64
Total Elapsed Time	670.58s	914.87s	1336.87s
FE Time	214.47s	262.55s	493.39s
FE CPU Time(user)	156.53s	214.36s	447.17s
FE CPU Time(system)	10.40s	10.80s	15.51s
FE I/O Time	47.54s	37.39s	30.71s
CM Time	40.04s	89.40s	250.69s
CM CPU Time	6.52s	17.14s	60.41s
CM Comms Time(Send/Get)	30.58s	68.91s	183.31s
CM Comms Time(News)	2.16s	2.50s	5.91s
CM Comms Time(Reductions)	0.78s	0.83s	1.06s

Table 3: Timings for individual based simulation, doing 1000 iterations with a carrying capacity of 4, 2 foxes per field and 4 rabbits per field at the start and 3 different grid sizes.

communications. It is therefore relevant to compare the performance of the two programs. Tables 3 and 4 shows the results of running both programs with 3 different sizes of world. The timings were done on the Connection Machine, using Prism, and in timesharing mode(ie with other processes running). Thus the timings should not be considered precise.

As we can see, when the grid size gets up to 64x64, giving roughly a 16 fold increase in the number of individuals being processed, the two programs run in roughly the same time, but for smaller simulations the spatially based model is the clear winner. The machine had 8192 processors available when running the program and the parallellism prevented a proportional increase in run time in both cases. Other points to note are:

1. The individual based simulation spends a lot more time doing comms than the spatially based simulation.
2. The amount of Front End(FE) time increases less quickly in the individual based simulation, and the amount of Connection Machine(CM) time is much greater than in the spatially based simulation.
3. The I/O times are unreliable since they are affected by other processes, how much network traffic there is etc...
4. A whole class of communications simply isn't used by the spatially based simulation.
5. The individual based simulation can have a lot of computation added without as big a hit as would happen in the spatially based simulation.

Grid Size	16x16	32x32	64x64
Total Elapsed Time	293.93s	570.93s	1442.37s
FE Time(total)	24.00s	65.23s	168.91s
FE CPU Time(user)	8.50s	41.77s	111.86s
FE CPU Time(system)	2.07s	3.89s	3.77s
FE I/O Time	13.45s	19.57s	53.28s
CM Time(total)	3.41s	8.97s	21.45s
CM CPU Time	0.55s	6.12s	18.21s
CM Comms Time(Send/Get)	0s	0s	0s
CM Comms Time(News)	0.78s	0.81s	1.18s
CM Comms Time(Reductions)	2.08s	2.05s	2.06s

Table 4: Timings for spatially based simulation, doing 1000 iterations with a carrying capacity of 4, 2 foxes per field and 4 rabbits per field at the start and 3 different grid sizes.

It seems then that although there is a sacrifice made in communications time, the individual based model has more scope for being extended before its run times get too long.

3 Adding competing strategies to the model

The next stage of the project involved adding competing migration strategies to the individual based simulation. The strategies used for foxes were:

1. **Greedy Foxes.** These foxes are most likely to choose the field with the largest number of rabbits in it when migrating.
2. **Shy Foxes.** These foxes are most likely to choose the field with the lowest number of foxes in it when migrating.
3. **Random Foxes.** These foxes migrate at random, as in the two previous programs.
4. **Dumb foxes.** These foxes are most likely to migrate to the field with the lowest number of rabbits in it.

Strategies 3 and 4 were used as controls — dumb foxes should die out very quickly, and random foxes should on the whole do worse than either shy or greedy foxes. Shy foxes' advantages are that they minimise the competition between them and other foxes for whatever rabbits there are in the fields they migrate to, and tend to gain an even distribution across the fields, thus they are less likely to wipe out concentrations of rabbits. Greedy foxes' advantages are that they maximise the number of rabbits they can catch, and can take advantage of concentrations of rabbits. One would expect that Greedy foxes would in general win out.

The strategies used for rabbits were:

1. **Clever Rabbits.** These rabbits are most likely to choose the field with the lowest number of foxes in it when migrating.
2. **Shy Rabbits.** These rabbits are most likely to choose the field with the lowest number of rabbits in it when migrating.
3. **Random Rabbits.** These rabbits migrate at random, as in the two previous programs.
4. **Dumb Rabbits.** These rabbits are most likely to migrate to the field with the largest number of foxes in it.

Strategies 3 and 4 are again used as controls here. The advantages of shy rabbits are that they tend to spread themselves evenly throughout the fields, and avoid forming concentrations which might attract greedy foxes. The advantage of clever rabbits is that they actively seek fields with the lowest number of foxes in them. One would expect clever rabbits to be the most successful.

All migrations are synchronised, and the animals can only look at or move to horizontally or vertically neighbouring fields in the grid. Thus all the strategies use only local information, and the information is only correct for the timestep previous to the current migration. In order to implement these strategies the following additions needed to be made:

1. A method of gathering information about neighbouring fields.
2. A way of indicating which strategy and individual is following.

The first was achieved by adding eight fields to each record, indicating the number of foxes and rabbits to the north, south, east and west of the field. Then, the number of foxes and rabbits in each field was read off into a 2d grid. These values were then read back to the relevant individuals to obtain the number of foxes in the neighbouring fields.

The second was achieved by adding an extra field indicating what strategy an individual follows. When the program is run, it asks for the proportions of the population to be assigned to each strategy.

4 Experiments with competing strategies

Two sets of experiments were performed with the competing strategies — one where all the rabbits moved randomly and the foxes had competing strategies. The other set had all the rabbits following a single (non-random) strategy, plus an experiment where both foxes and rabbits start out with the proportion of the population following each strategy being roughly equal. Unless otherwise stated, all the experiments were done with a grid size of 32x32, a carrying capacity of 4, starting out with 2 foxes per field, and 5 rabbits per field. Usually the experiments were run for 2000 iterations.

4.1 Experiments with competing fox strategies, and randomly moving rabbits

There were 5 of these experiments in total, 1 where the proportion of the foxes following each strategy is equal, and the rest where each strategy in turn was initially the dominant one. All the experiments were run on the EPCC's Connection Machine.

4.1.1 Experiment 1: Equal initial proportions for each strategy

This experiment was run for 2000 iterations. Three runs were done, with the same inputs, to ensure that the behaviour was not a fluke of the random number generator. A graph of how the populations vary over time is produced in Figure 5. A graph of the proportion of the population following each strategy is produced in Figure 6. These graphs are from the third run, the other 2 runs produced similar results.

As we can see, over time, firstly the dumb foxes die out as expected, then the shy foxes, and finally the random foxes. The greedy strategy seems to be successful in these conditions. Although, in this run, the shy foxes died off before the random movers, in the other runs for this experiment it was the other way round, and in the other experiments this is confirmed unless one is initially dominant over the other.

4.1.2 Experiment 2: Random foxes initially dominant

This experiment was run for 200 iterations, and 3 runs were done as before. Figures 7 and 8 show the variations in the populations, and in the proportions of the foxes following each strategy respectively.

The main thing to note about the graphs is, again, the success of the greedy strategy over the others. The dumb strategy dies off first as expected, and given the initial dominance of the random foxes it is not suprising that the shy foxes die off next.

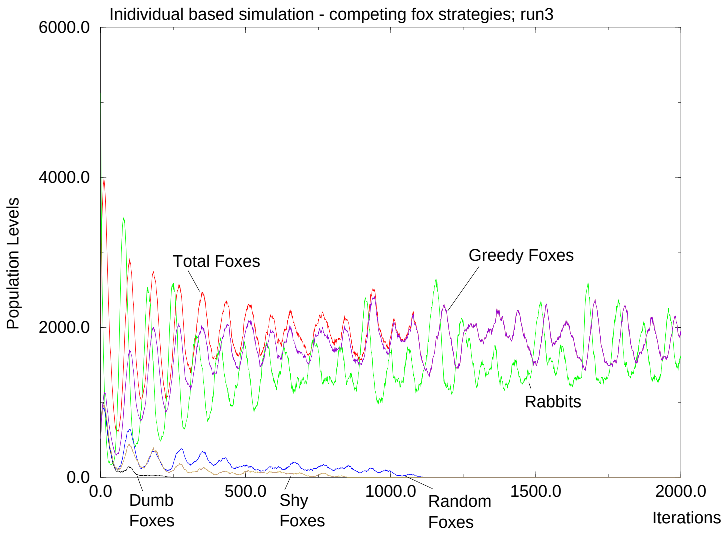


Figure 5: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.1, experiment 1.

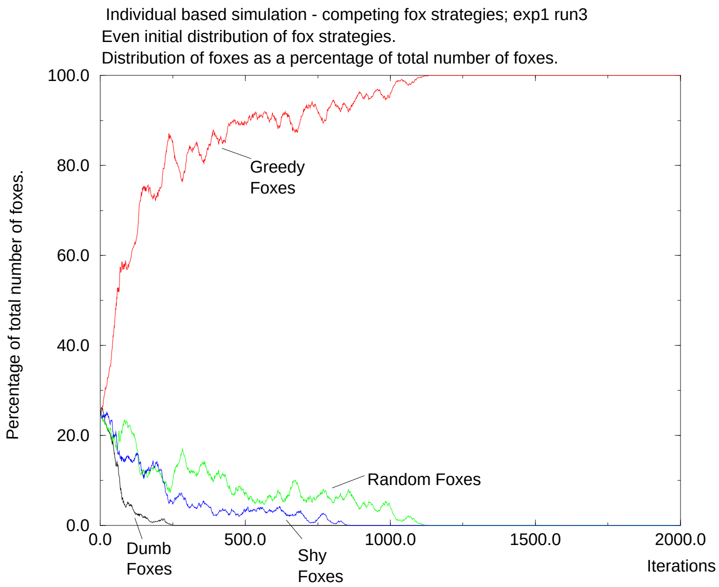


Figure 6: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.1, experiment 1.

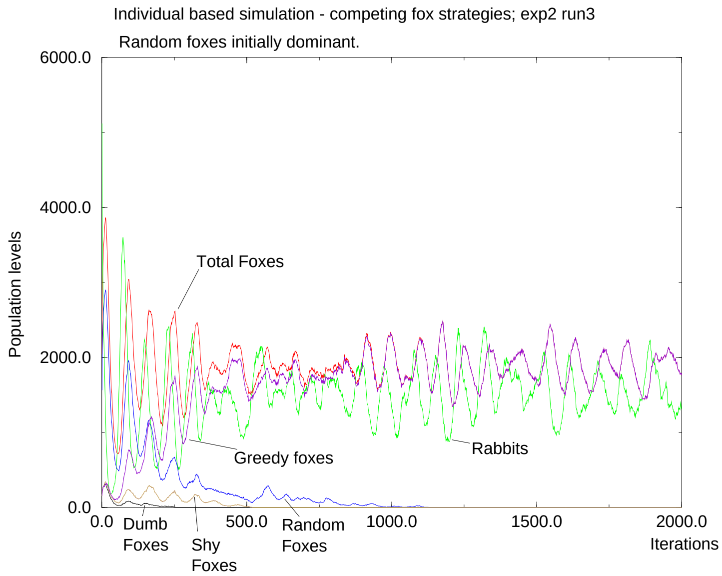


Figure 7: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.1, experiment 2.

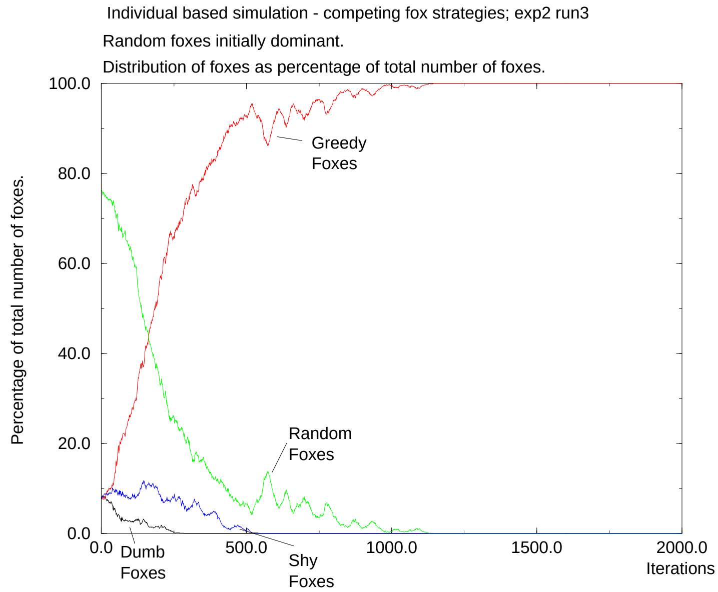


Figure 8: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.1, experiment 2.

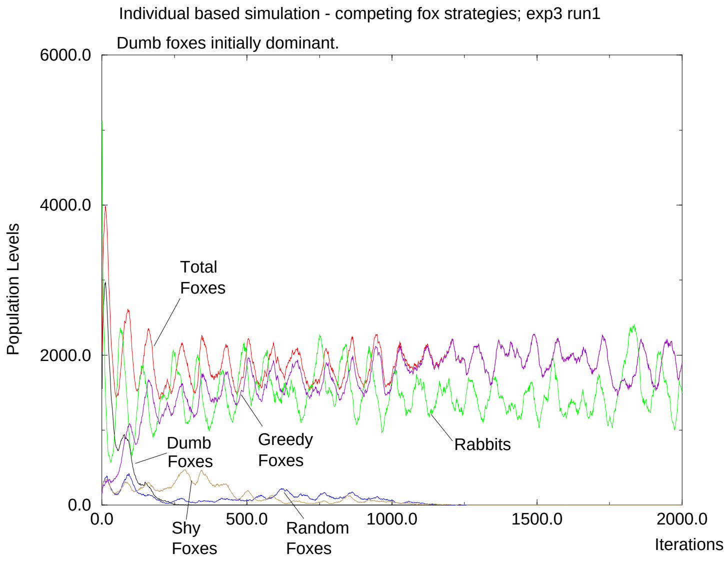


Figure 9: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.1, experiment 3.

4.1.3 Experiment 3: Dumb foxes initially dominant

This experiment also ran for 2000 iterations, and was run 3 times. The graphs in Figures 9 and 10 are from the 1st of these runs.

Again, the greedy foxes quickly achieve dominance, but notice also that the dumb foxes, despite their initial dominance die off well before the shy and random foxes. The latter two seem to die off at roughly the same time.

4.1.4 Experiment 4: Shy foxes initially dominant

For this experiment, 3 runs were done of 2000 iterations, plus an extra run for 4000 iterations. The extra run was done because in all 3 earlier runs, the shy foxes were still around in significant numbers at the end of the run, and it seemed that it would be worthwhile seeing if this held out over 400 iterations. The graphs in Figures 11 and 12 show the population variations over time for 2000 iterations and 4000 iterations. Figure 11 is from the first of the 3 runs at 2000 iterations. Figure 13 shows how the proportion of the fox population following each strategy varies in the 4000 iteration run.

As we can see, in the 4000 iteration run, the shy foxes finally die out. This suggests that the shy foxes are a better strategy than either the random movers or the dumb foxes, since the other two die out much more quickly if they are given initial dominance. Nevertheless, the greedy foxes end up winning again here, confirming that this strategy is best when the rabbits are all random movers.

4.1.5 Experiment 5: Greedy Foxes initially dominant

In this experiment 3 runs were done all of which were 2000 iterations. The graphs in Figures 14 and 15 were taken from the first run.

As we can see the greedy foxes quickly go from a position of dominance to being the only foxes left, again confirming the success of this strategy. The other three strategies quickly get wiped out, again with the dumb foxes going first, then the shy and random foxes.

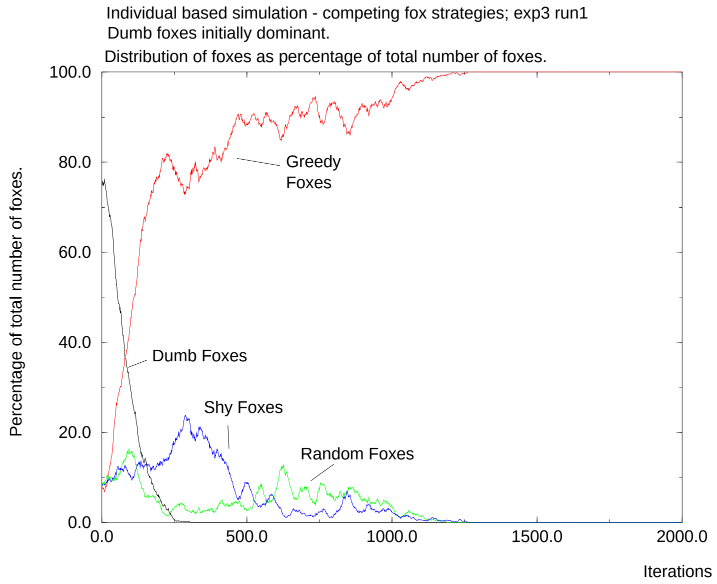


Figure 10: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.1, experiment 3.

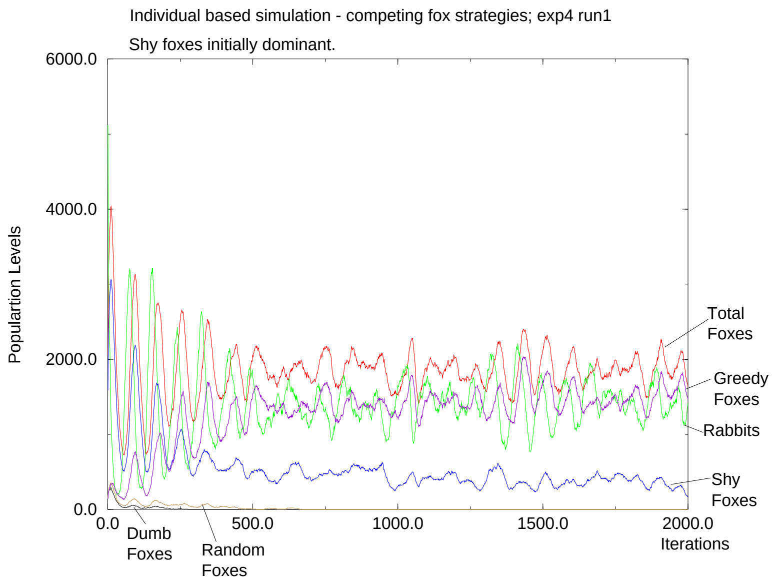


Figure 11: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.1, experiment 4.

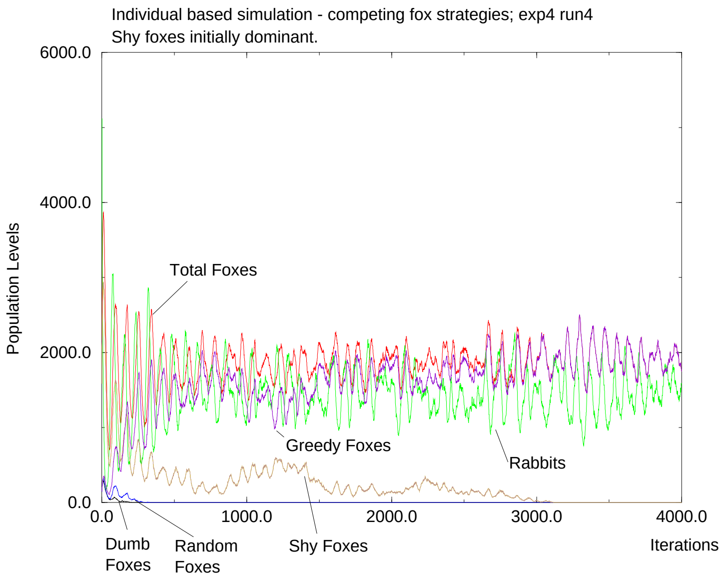


Figure 12: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.1, experiment 4.

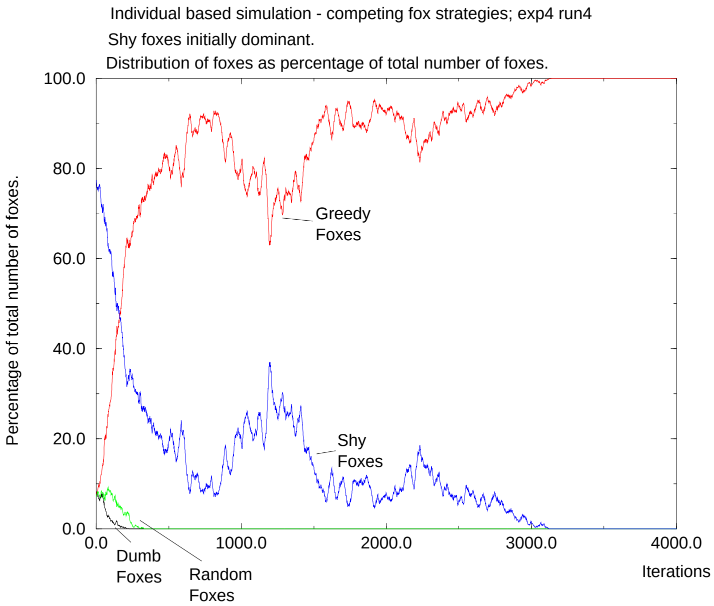


Figure 13: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.1, experiment 4.

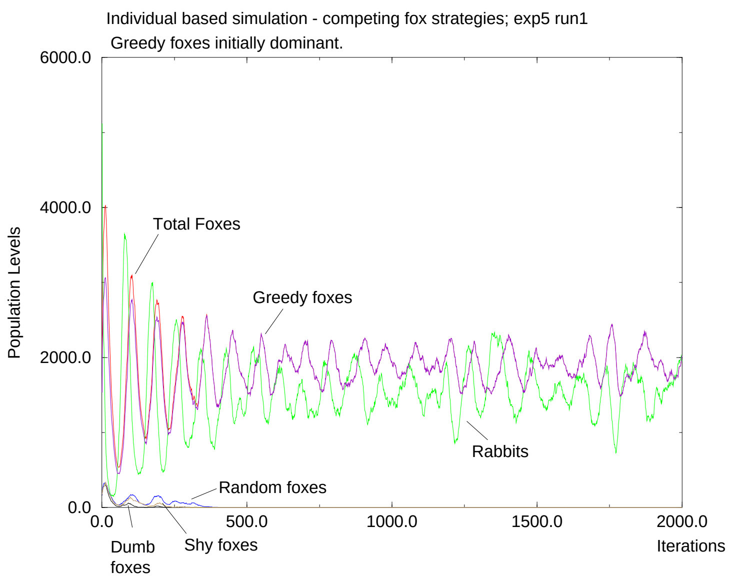


Figure 14: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.1, experiment 5.

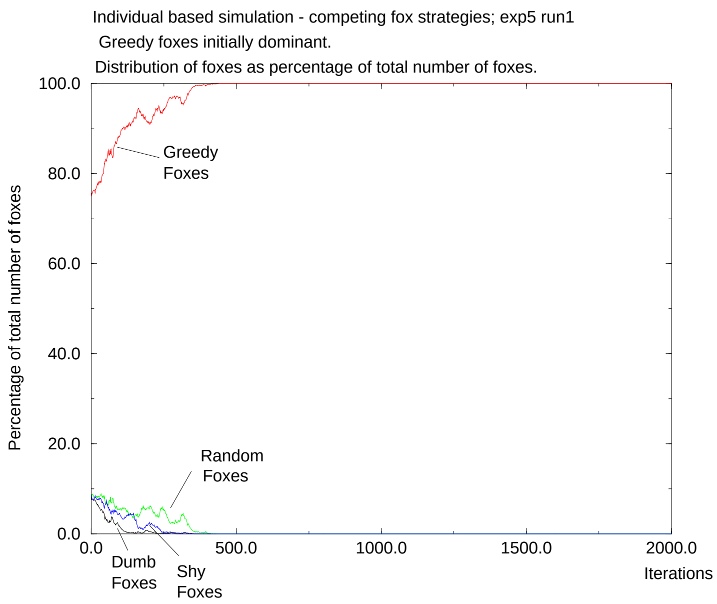


Figure 15: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.1, experiment 5.

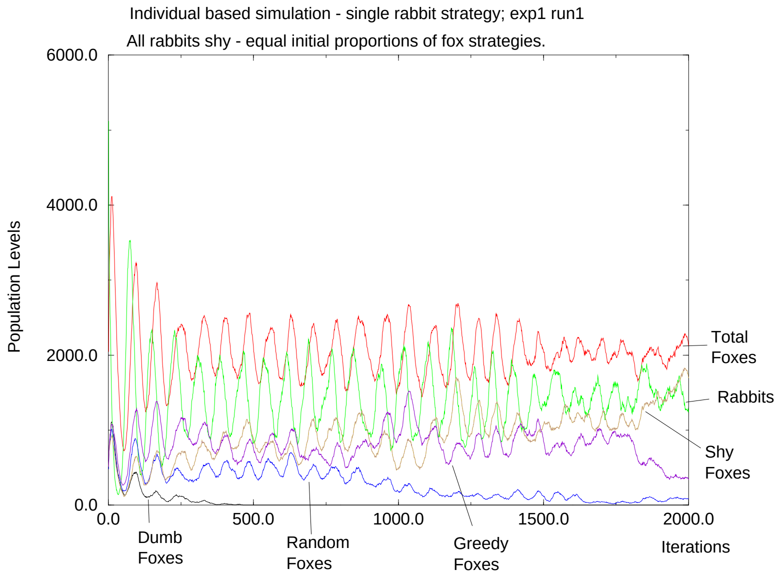


Figure 16: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 1.

4.2 Experiments with rabbits following strategies.

There were 4 experiments in total which had the rabbits moving non-randomly, 3 with all the rabbits following a single strategy, and one where initially equal proportions of the rabbit population follow each strategy. For all these experiments, initially equal proportions of the foxes follow each fox strategy.

4.2.1 Experiment 1: All rabbits shy

In this experiment 3 runs were done with 2000 iterations, and a run each at 4000 and 8000 iterations. The reason for this is that in all runs, the greedy and shy foxes battle it out for supremacy. Neither of them gets wiped out, and no sooner does one get ahead than the other starts to fight back. Running the experiment for 800 iterations showed no deviation from this pattern. Figure 16 and 17 show the variations in population over 2000 and 8000 iterations respectively. Figure 18 shows how the proportions of the fox population following each strategy vary over time.

Note that although neither the greedy nor shy foxes manage to get a permanent lead, the shy foxes seem to be in the lead for more of the time than the greedy foxes. It appears that when the rabbits are shy, shy foxes gain a clear benefit from this. This is probably because both shy foxes and shy rabbits spread themselves fairly evenly over the fields — greedy foxes are no longer in as strongly as advantageous position as when the rabbits were moving randomly, and cannot find large concentrations of rabbits to take advantage of. This has the makings of an ESS in it, but since the distribution seems to oscillate a lot it's hard to say what the relevant proportions are. It would seem plausible that they might be close to the average proportions over the 8000 iterations of the final run. Finally note that the dumb foxes still die off first.

4.2.2 Experiment 2: All rabbits clever

Three runs were done, each for 2000 iterations, and a fourth run was done for 4000 iterations. The graphs of how the populations vary in the 1st run and the 4th run are given in Figures 19 and 20. The graph of how the proportions of the fox population following each strategy varys is given in Figure 21.

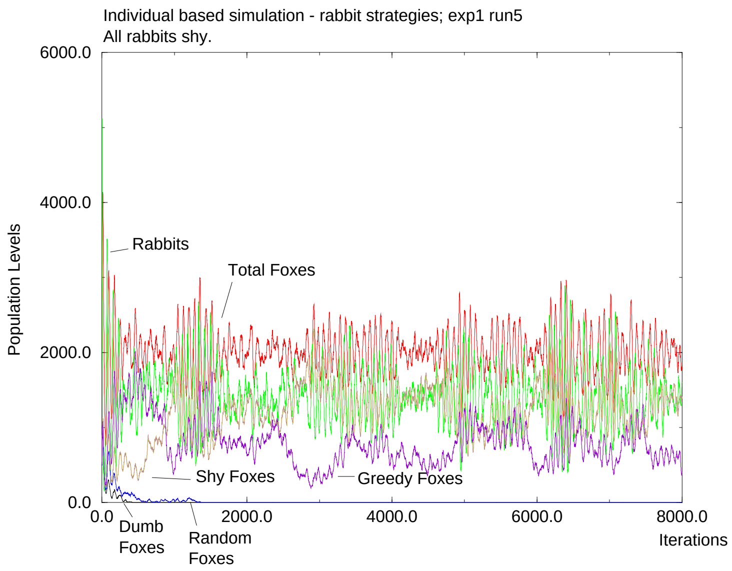


Figure 17: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 1.

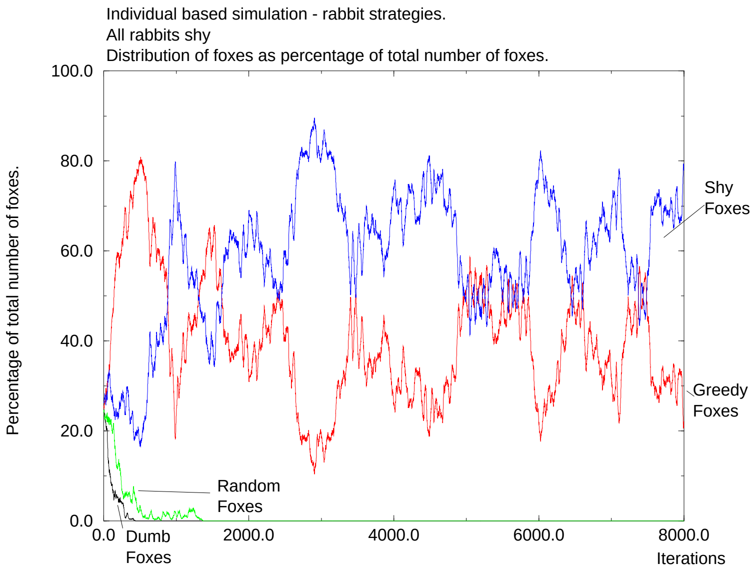


Figure 18: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.2, experiment 1.

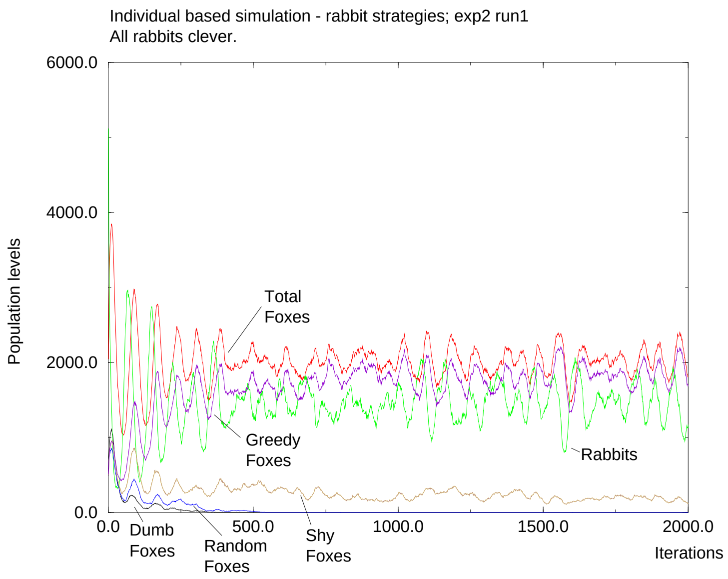


Figure 19: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 2.

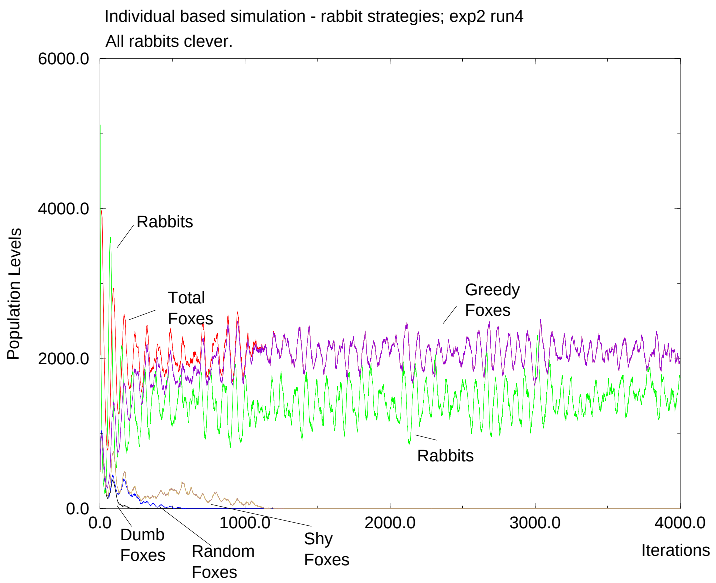


Figure 20: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 2.

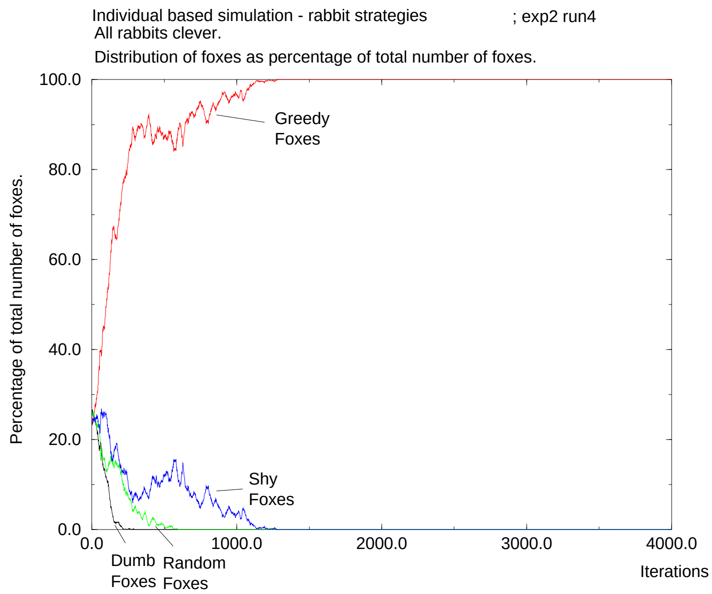


Figure 21: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.2, experiment 2.

This time, all but the greedy foxes are wiped out in the long run, although the shy foxes hang on for a while. The strategy the rabbits follow will lead to concentrations of rabbits appearing more often than the shy strategy does, hence the shy foxes do not have much advantage in this situation.

4.2.3 Experiment 3: All rabbits dumb.

In this experiment, all the rabbits followed the dumb strategy. 5 runs of the experiment were done, 3 at 2000 iterations, 1 at 4000 iterations and 1 at 8000 iterations. The graphs of the variations in population for the 1st and 5th runs are given in Figures 22 and 23. The graph of how the proportions of the fox population following each strategy vary is given in Figure 24.

As we can see, the shy foxes and greedy foxes again battle it out for supremacy, but notice that the general size of the populations is lower, and that the populations go extinct in the 8000 iteration run. This is not unexpected; what was unexpected was that it didn't happen in the earlier 4000 iteration or 2000 iteration runs. Nevertheless this suggest that having the rabbits follow the dumb strategy will lead eventually to their extinction, as expected. Before the extinction occurs, both the greedy and shy foxes seem to have an advantage, though since the rabbits are in effect offering themselves to the foxes this is not too surprising — the shy foxes are minimizing the competition for the rabbits offered to them and the greedy foxes are maximizing the number of rabbits offered.

4.2.4 Experiment 4: Equal initial proportion of rabbit population following each strategy

Here the experiment lets the rabbit strategies compete in the face of competing fox strategies. 4 runs were done, 3 at 2000 iterations and 1 at 4000 iterations. Graphs for the variations in the populations in run1 and run4 are given in Figures 25 and 26. Graphs for the variations in the proportions of each population following each strategy are given in Figure 27.

As we can see, eventually the clever strategy of the rabbits, and the greedy strategy of the foxes become the sole strategies that the two populations follow. This is in line with the intuitions that foxes which actively seek the fields with the largest number of rabbits, will do best, and that similarly rabbits which seek the fields with the lowest number of foxes will survive best too.

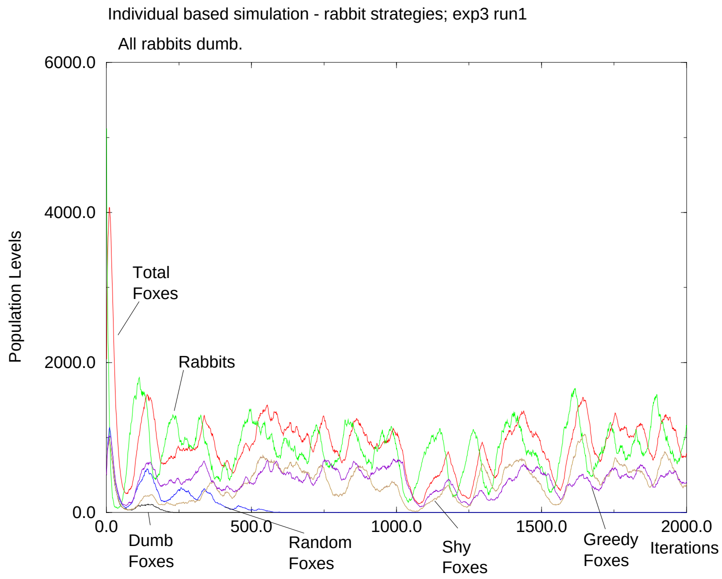


Figure 22: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 3.

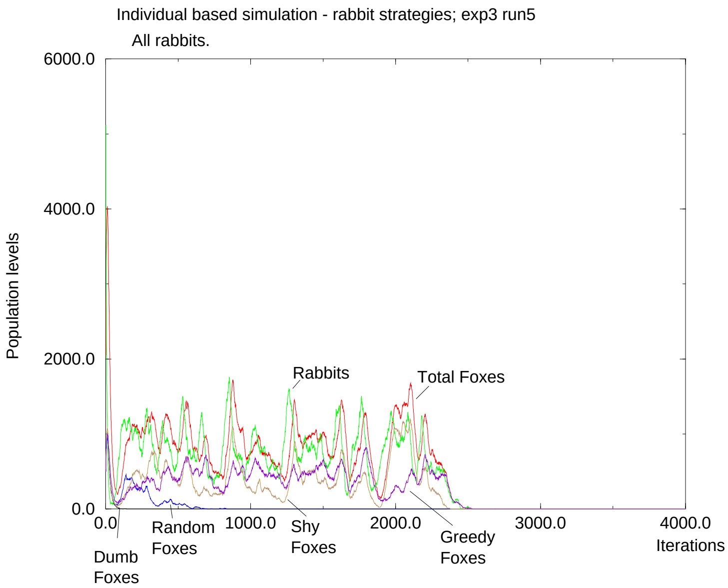


Figure 23: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 3.

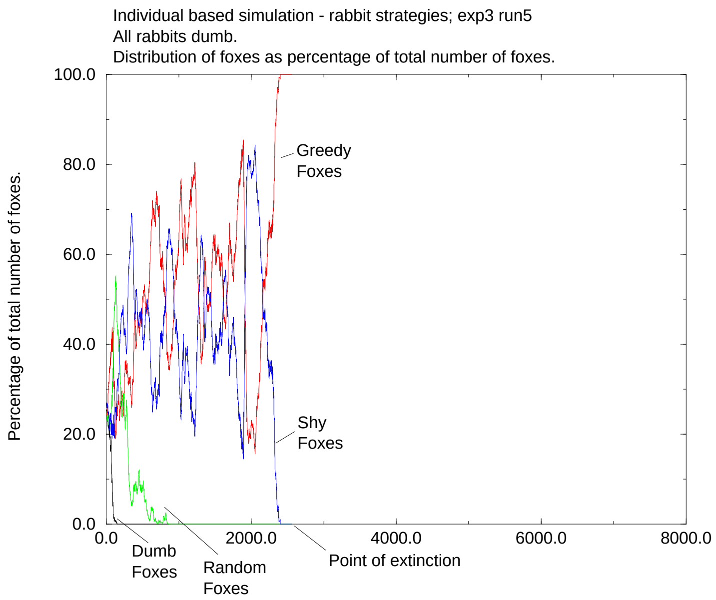


Figure 24: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.2, experiment 3.

5 Conclusions

The individual based simulation has allowed the creation of a predator-prey model were individual animals follow their own migration strategy, albeit with a fairly large communications overhead. However this overhead increases less than proportionately with the amount of data to be processed, and as such could allow larger simulations to run in times better than those of a similar spatially based simulation. It also allows a more complex model to be built without a big hit on the performance.

The experiments show that the greedy and clever strategies are the strongest, given that all other things are equal — though when all the prey are shy, shy predators benefit, and it looks as if an ESS between greedy and shy predators arises here. Where both foxes and rabbits are following competing strategies it seems as if the clever and greedy strategies will eventually completely displace the others. None of this should be too surprising, although the case where all the rabbits are shy seems quite interesting. However one important element of evolution is missing from this program — mutation. If one were to allow a small chance for a fox to give birth to another fox following a different strategy one may find that this alters the findings given here — at the moment a greedy fox can only give birth to another greedy fox, likewise for the rest of the animals and strategies.

6 Extensions to the project

As it stands the program is rather simplistic — although it captures the essence of survival of the fittest, it allows no mutations, and every animal sticks rigidly to one strategy. There are many extensions one could add. Indeed this a very open ended area — one can add new features, increase the complexity or add more realism almost indefinitely, the power of the available computings being the main restriction. As a start the following additions could lead to a model which has all the main characteristics of evolution in it:

1. Add mutations. At its simplest this would simply mean allowing animals to have offspring following different strategies from their parent. It would be interesting to see if this, by itself, changes any of the above results.

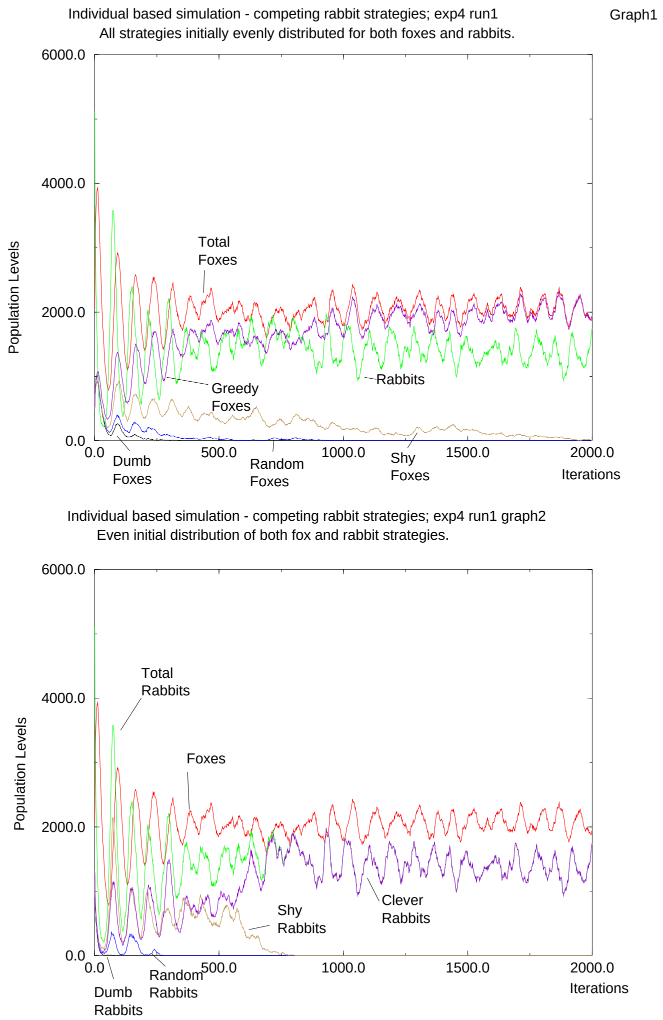
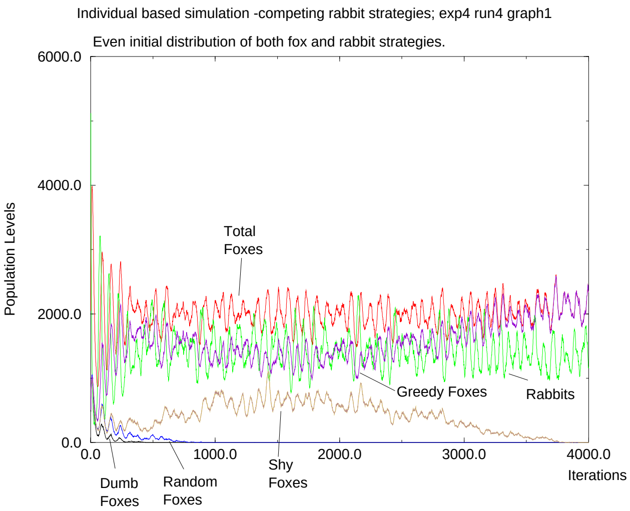


Figure 25: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 4.



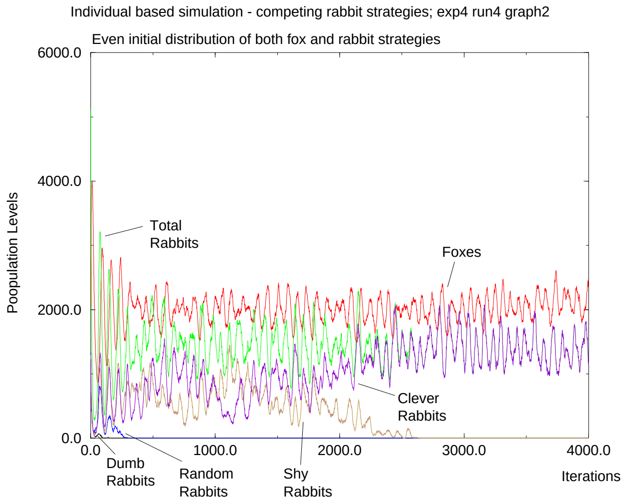
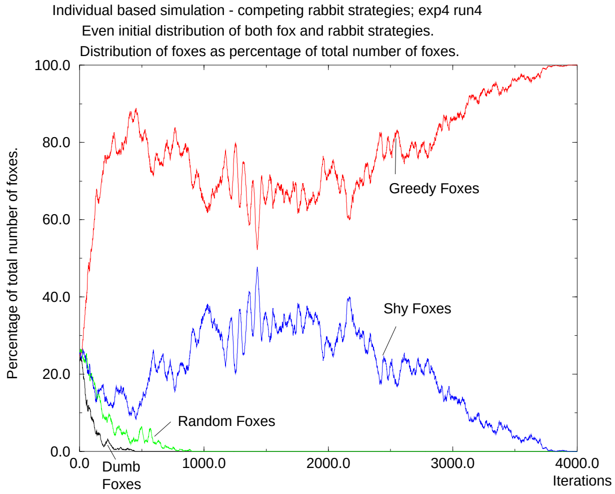


Figure 26: Graph showing how population levels of the foxes and rabbits vary over time in the individual based simulation, with competing strategies. See section 4.2, experiment 4.



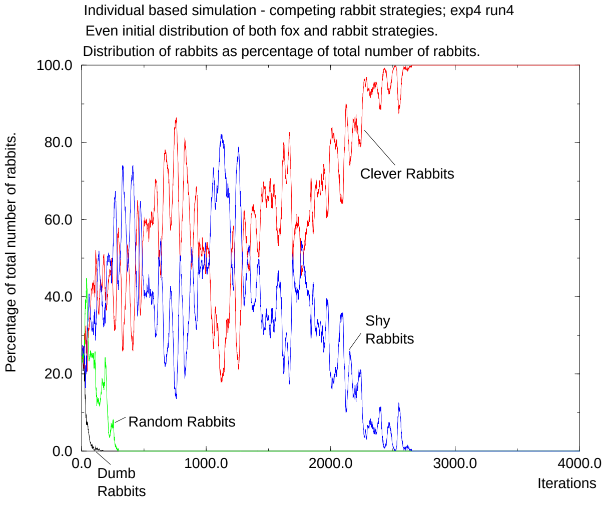


Figure 27: Graph showing how the proportions of the fox population following each strategy vary over time. See section 4.2, experiment 4.

2. Add trade-offs to the choices an animal might make. At the moment on each cycle, the animals will give birth or die, then predations will occur, then migrations would occur. It would seem more realistic if instead there was a probability of each animal either migrating, if it's a fox to try to catch a rabbit, or doing nothing. Each action would incur some sort of cost — you could have an energy level for each animal, which decreases over time, and which is further decreased by migrations, births and predations. Successful predations would of course increase the energy, and the lower an animal's energy level, the greater the likelihood of it dying. The program would then have a two stage cycle; first decide which animals die, then decide whether the survivors give birth, migrate, predate or do nothing. Having the probabilities of migration, predation, doing nothing and giving birth mutate when a birth takes place would then allow evolution to determine the amount of time an animal spends doing these things, and hence the best balance for survival. Indeed, the collection of probabilities would become very simple genes.
3. Add mating to the model, so that you can get better mixing of the genes.
4. Have a seasonal food supply for the prey — you would probably want a food supply of some sort in order to implement item 2 for prey. Making it seasonal would add another element of the real world into the simulation.
5. Add an evolveable strategy for when an individual migrates. This might allow some otherwise unsuccessful strategies in choosing what to do to evolve successful variants.

The foregoing additions would have all the main elements of simulated evolution — mutation, natural selection and mixing of genes. At the moment the simulation only really has natural selection built into it.

References

[Dawkins, 1978] R. Dawkins. *The Selfish Gene*. Oxford Univeristy Press, paperback edition, 1978.

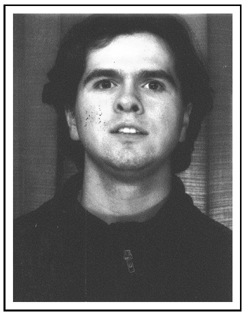
[Lotka, 1925] A. J. Lotka. *Elements of Mathematical Biology*. Dover, reissued 1956 edition, 1925.

[Maynard-Smith and Price, 1973] J. Maynard-Smith and G. R. Price. The logic of animal conflicts. *Nature, Number 246*, 1973.

[Smith, 1991] M. A. Smith. Using massively parallel supercomputers to model stochastic spatial predator-prey systems. *Ecological Modelling, Issue 58, pages 347-367*, 1991.

[Smith, 1993] M. A. Smith. Dynamic load-balancing strategies for data parallel implementation of reaction-evolution-migration systems. *International Journal of Modern Physics, Vol. 4 Number 1*, 1993.

[Volterra, 1926] V. Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature, Number 118*, 1926.



James Hammerton recently graduated with a joint honours degree in Artificial Intelligence and Computer Science at the University of Edinburgh. After the EPCC Summer Scholarship Program he will be moving to the University of Birmingham to start a PhD in Natural Language Processing.
This project was supervised by Mark Smith (EPCC).