Problem Introduction

In this paper we seek to investigate the effects of different breeding strategies on the performance of an evolutionary algorithm.

The strategies we explore can be split into two different categories, those that effect when an agent reproduces and those that decide which two agents are selected to reproduce. The different strategies that define when an agent reproduces are well explored and defined in evolutionary biology whereas the strategies we use for partner selection are less based on natural systems. This means that our areas of investigation will be see what partner selection strategy is most beneficial for each "when" strategy.

To measure the performance of the system, each agent has a "fitness" which increases whenever they collect a piece of food, of which a fixed number are placed in the environment. We define the performance of a system as the average fitness of all agents in the world.

The Model

We built our model on top of one provided by Simon Lynch, changing little except who is selected to breed and when it happens. The original model included functionality to have predators hunt agents but this was removed as it was deemed superfluous to the problem we wished to investigate.

Agents exist in a continuous world (a torus) and have their behaviour defined by a set of rules which are randomly generated for all the agents that are created at setup. Rules for agents that are created during run time are generated from a mix of two other agents rule sets. A number of pieces of food are also placed in the environment, when an agent occupies the same area as a piece of food, the food is consumed and the agent's fitness is increased. If an agent's fitness equals zero, the agent is killed.

Rules

Rules are represented as having two parts, a sensor and an effector. When a particular sensor is evaluated to true, the agent performs the effector associated with it.

The rules are defined as:

```
set *sensors*
    table:from-list
    [["000"  "target-fwd"   ]
     ["001"  "target-left"  ]
     ["010"  "target-right" ]
     ["011"  "true"         ]]

set *effectors*
    table:from-list
    [["00"  "move-fwd"   ]
     ["01"  "turn-right" ]
     ["10"  "turn-left"  ]
     ["11"  "turn-180"   ]]
```

This will create a genome of the form:

```
"01011101001101110111"
```

Which, when translated into a phenome creates:

```
"[ ifelse (sensor.target-right) [
effector.turn-180 ]

[ ifelse (sensor.target-left) [
effector.move-fwd ]

[ ifelse (sensor.false) [
effector.turn-right ]

[ effector.nop ]]]]]"
```

If no sensors evaluate to true, then no action is taken.

These rules can then be perceived as good or bad; if an agent's rules tell it to walk away from food then it will eventually starve to death and remove itself from the gene pool. If its rules tell it to go towards food it will increase its fitness and have a chance to pass this good rule to a new agent.

Evolving

To cross over two genomes and create a totally new one we iterate over the length of a genome and select at random which of the parents to take a rule digit from. The algorithm is biased towards whichever genome it took a digit from last, so the resultant genome will likely have sequences in common with its parents. The frequency at which the algorithm swaps between the parent genomes is exposed and called #xprob.

```
to-report genome.cross-over [#genome-a #genome-
b #xprob]
  let #res ""
  let #tmp []
  let #n 0
  repeat (length #genome-a)
  [ if (roll-dice #xprob)
    [
      set #tmp #genome-a
      set #genome-a #genome-b
      set #genome-b #tmp
    ]
    set #res (word #res (item #n #genome-a))
    set #n (#n + 1)
  ]
  report #res
end
```

The new genome will then have a random chance to be exposed to mutation. This algorithm loops through the genome and has a small random chance to cause a rule digit to become its opposite.

```
to-report genome.mutate [#genome #genes #mprob]
  ;; genes must be a list of allowed genes, eg:
[0 1]
  let #res #genome
  let #n 0
  repeat (length #genome)
  [ if (roll-dice #mprob)
    [ set #res (replace-item #n #res (word
(one-of #genes)))
    ]
    set #n (#n + 1)
  ]
  report #res
end
```

Breeding Periods

We explore three different types of breeding periods in this investigation, these are:

Polycyclic: Polycyclic organisms reproduce intermittently throughout their lives.

Semelparous: Semelparous organisms reproduce only once in their lifetime and often die shortly after. They generally have many offspring at once. Agents in our model can have up to six offspring at once. This value maintains a constant population of agents.

Iteroparous: Iteroparous organisms reproduce in cycles (e.g. annual or seasonal) and commonly survive to reproduce in successive seasons.

Each of these strategies was very simple to implement. For Polycyclic we only allowed an agent to reproduce if a random number within the range zero to a number prescribed by us equals zero.

For Semelparous we allowed agents to breed only once by triggering a flag inside them to true the first time they reproduce, when this is triggered the agents are then killed off before ten more ticks have been executed.

To implement Iterparous reproduction, we gave every agent a number to represent its season. Every time this agent is asked if it can breed, it mods the current tick number of the program with this number, if this equals zero, it can reproduce.

Parent Selection

For this paper, we investigated the effects of four different selection strategies. These are:

Elitism: Two random agents with fitness above a certain threshold are selected as the parent.

Proximity Elitism: Same as elitism but the two agents must be within a certain radius of one another.

Random: Choses two random parents from the whole pool of agents.

Ladder: The agents are sorted into ascending fitness and a random number between zero

and the sum of all agent fitness. Then all agents are looped through, while keeping track of the cumulative fitness. If the random number falls between the current total fitness and the total fitness plus the fitness of the current agent, this agent is selected as a parent. This process is then repeated for the other parent.

Predictions

We predict that elitism will quickly increase the fitness in all the different breeding periods. The strategy is obviously strongly biased towards the better performing agents and completely ignores the weak agents. The lower we set the threshold for "elite" agents the slower the increase in average fitness. Although, this may end up with more fit agents in the long run as the larger pool of agents would eliminate any biases caused by agents with worse rules but better circumstances.

Proximity elitism would produce similar results to elitism, with average fitness increasing over time. However, the increased difficulty in finding a mate coming from the proximity stipulation means that this increase should be much shallower than simple elitism.

Random breeding should also produce an increase in average fitness over time. We predict that this increase will be the slowest of any of the different strategies because it has no biases as to the parents it picks. However, it will still increase over time as the weakest agents die off and not get a chance to reproduce.

The ladder selection strategy will likely also increase average fitness over time. This increase will most likely be slower than elitism's increase because of its wider selection of agents. The strategy has a much more likely chance of selecting a fit agent but it does not totally negate the possibility of picking weak agents.

As for the breeding periods, the constant nature of Polycyclic, coupled with the fact that fit agents can reproduce multiple times strongly implies that it will produce a constant steady increase in fitness quicker than the other period strategies.

For Semelparous agents we predict that it won't show any string evolutionary trend as the fittest agents can only reproduce a limited amount of times.

Finally, we predict that Iterparous agents will show the same evolutionary trends as Polycyclic agents but over a larger period of time as we limit not how many times they can reproduce, but when they can do it.

Experiments

Set Up

All experiments were performed in the NetLogo environment that the model is built in. NetLogo provides several different ways to take information from a model and display it as data. The most useful of these tools is probably the Behaviour space, which allows you to create several different "Monte Carlo" style runs of your model and output the data as comma separated values. However, because of the complexity of out model and the amount of agents running at once this method was much too slow to test our model in. Instead, we ran the models manually in the normal NetLogo environment and recorded the results of each separate run.

For each run we recorded the amount of agents at the end of a run, the average fitness of all these turtles and the fitness value of the fittest agent. All runs are allowed to execute for 2000 ticks.

Data

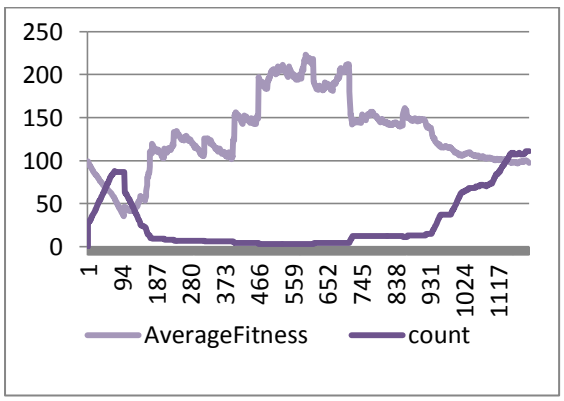
For the test we used the following data:

- A population of 30 agents in a 20 by 20 cell world.
- 70 pieces of food throughout the world.
- Genomes are swapped during breeding 1 in 8 times.
- Genomes mutate 1 in 18 times.
- Food increases agent's fitness by 15.
- Moving cells decreases an agent's fitness by 1.
- Agents can see the contents of cells up to 6 cells away.
- When in proximity elitism, two parents must be less than 5 cells distance from one another.

Observations

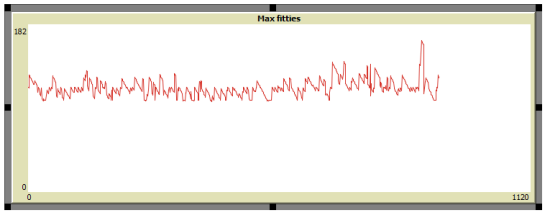
The deaths of agents show a working model.

In many different runs of our model, it is clearly observable that a large uptick in average fitness occurs at the same time as many deaths. This is down to agents with zero fitness dying off, taking their fitness out of consideration for the average and only leaving the fit agents.

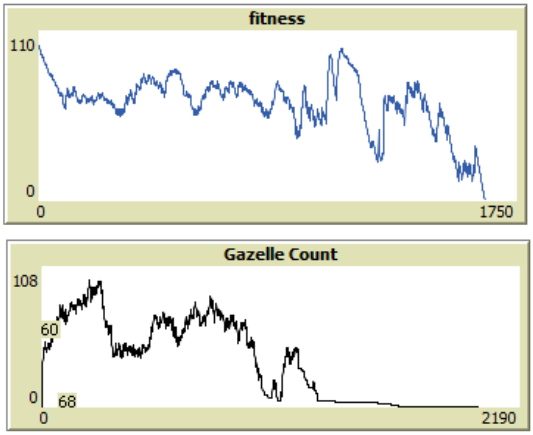


Semelparous agents struggle to evolve.

Whenever a Semelparous reproduces, it genes are taken out of the gene pool forever and it will die shortly after. This makes it very difficult for advanced genes to rise to the top in any of the breeding strategies. And in many cases we saw that the whole population would die long before the end of the test.

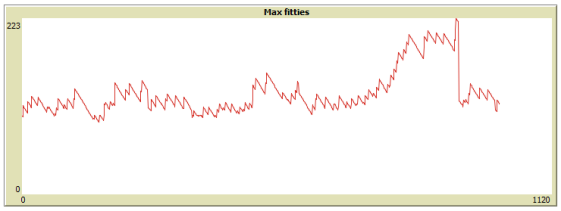


Fitness does not increase in semelparous societies (Proximity Elitism)



The degradation of fitness in a Semelparous society leads to the death of the entire population.

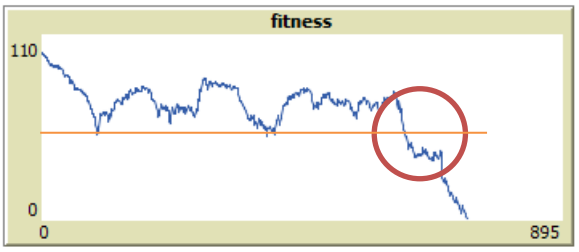
The deaths of the fittest agents are easy to locate in a graph of the fitness value of the fittest agent. Large drops, like the one in the graph below are caused by the fittest agent dying shortly after it has reproduced.



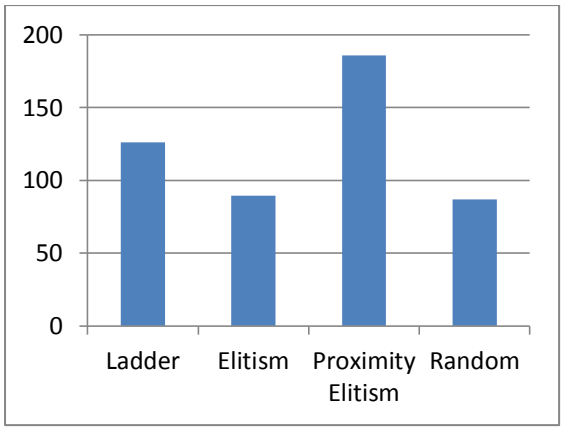
Large drops in max fitness when the fittest dies after breeding

Fitness has an event horizon in Semelparous societies.

In semelparous societies fitness has a “point of no return”, a value that if it should fall below, the fitness of a society will only continue to degrade until the population dies out.



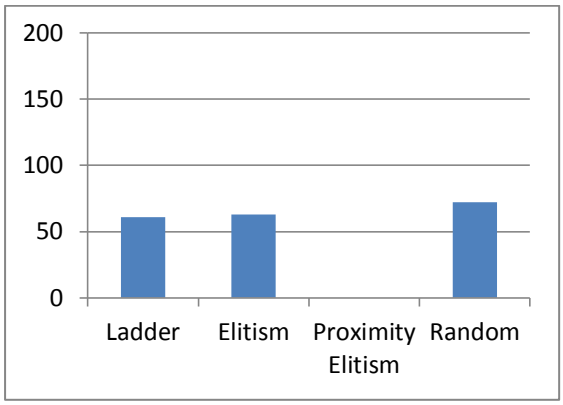
Polycyclic Performance



Polycyclic societies performed best across all the different breeding strategies beating Semelparous and Iterparous in its average fitness.

Out of the different strategies, Proximity elitism performed the best which is surprising given its stricter rules. The ladder strategy is the second lowest, with the elitism and random strategies both having similar low results. The low results of elitism are very surprising as we predicted that the strong bias it has towards fit agents would help it produce a large result.

Semelparous Performance

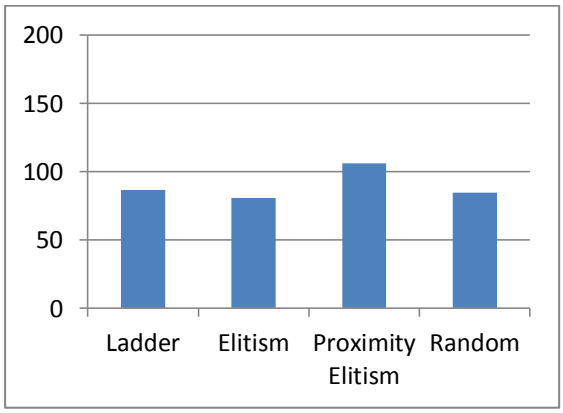


Semelparous had the lowest fitness values for each breeding strategy out of the three breeding periods. This is consistent with the predictions we made that the way it culls its fittest agents would make it an incredibly slow evolver.

The amount evolved is very similar across all the different strategies explored except for proximity elitism in which all the agents died during the runs we performed. This is more in keeping with the predictions we made about proximity elitism and how its strict rules for breeding would make it a worse performer.

As for the other strategies, the similarity of their results makes it hard to draw any concrete conclusions about their relative performance.

Iteroparous Performance

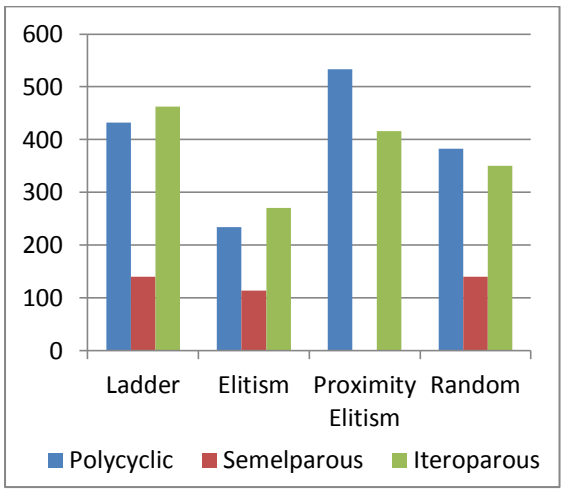


Iteroparous breeding produces much lower results than polycyclic, but ones that display the same trends between strategies.

Proximity elitism performs the best, then ladder, then similar low results for the elitism and random strategies.

Max Averages

For each run we also recorded the fitness value of the fittest agent that existed in the world. These values tell do not follow the same trends as the average fitness.



Again polycyclic and iteroparous agents both perform better than semelparous by a large margin. However, for the ladder and elitism strategies, iteroparous performs better than polycyclic. Even with these interesting results, it is more correct to draw conclusions from the average fitness as max fitness is much too liable to be effected by the random circumstance of an agent more than the model behind that agent.

Evaluations and Conclusions

From our tests it is very evident that semelparous reproduction is detrimental to the evolution of an organism. But many real life organisms possess these traits, so how can they survive in reality? We conclude that these animals first evolved into a state that would allow them to easily maintain a high population or existed in an environment that changed very little before evolving the traits of semelparous reproduction.

As for polycyclic and iteroparous strategies, both showed a constant uptick in fitness. Deciding if either is better than the other is unfair to do in this paper, as both would be suitable in different situations. Iteroparous for example would be a much better choice for ecosystems with a low amount of resources as the frequency of breeding is reduced. Polycyclic would perform better in environments where lifespans are shorter, mates sparser or where there are a large amount of predators.

Out of the breeding strategies it is hard to draw strong conclusions with a small set of rules and only one way of measuring their effectiveness. However, it is obvious from our tests that proximity elitism performs extremely well. Proximity elitism constantly performing better than elitism initially seems backwards, if you want to pick one of the best agents then surely picking out of all the agents improves the chances that you will get a strong agent? The answer is that the strong agents are probably close together anyway, one lucky group of agents will have been randomly placed into an area with lots of food and can then use one another to immediately breed strong agents into their group. While these agents don't have to have good rules to perform well, any bad rules they have will eventually be averaged out by breeding.

Future work

The model currently has only one set of rules and one way of judging the performance of those rules. To really get a sense of how the different breeding strategies are performing we would have to introduce separate sets of rules, the performance of which do not affect one another. An obvious way of doing this is to introduce predators and rules for avoiding these predators. With rules such as these it

would then be possible to evolve agents that were good at one thing but bad at others (e.g. good at finding food but bad at avoiding predators). In these situations I believe we would see a better evolutionary performance by the ladder strategy. Because the elitism strategy only has one way of deciding who is fit (who gets the most food) then it is not taking into account who is good at avoiding predators. The ladder technique does not ignore agents who are doing badly, it only picks them less. So it does not disregard agents who are bad at finding food but good at avoiding predators like an elitist strategy would.

Conclusion

The tests we have performed here are extreme simplifications of real world situations, and one could spend all day adding pieces of minutiae to the model. However the results we have gathered here do give us an overview of the performance of different breeding strategies used in nature and strategies used in evolutionary learning to maximise fitness in agents generated at run time.

For polycyclic and iteroparous agents we have proved that they lend themselves well to evolution and suggested real life situations that they would work best in. Semelparous reproduction definitely raised a lot more questions, as its benefits were a lot less obvious, however we suggested that it too has a place in the real world but perhaps might be the result of evolution rather than something that allows it.