Evolving Strategies for the Prisoner's Dilemma

Project Demonstration

Andrew Errity

99086921

26th May 2003

What is the Prisoner's Dilemma?

Two prisoners are placed in separate cells, with the aim of getting one prisoner to implicate the other. Each prisoner is given the option to **defect** against the other, by giving evidence against them, or to **cooperate** and withhold evidence.

- If both prisoners defect (give evidence) then the judge, in no doubt over their guilt, will send them both to prison for 3 years.
- If both prisoners cooperate (don't give evidence), then the judge, with less clear indication of guilt, will send them both to prison for only 1 year.
- If one prisoner defects and the other does not, the judge will take this as a clear sign of guilt, allowing the defector (evidence giver) to walk free whilst sentencing the other prisoner to 5 years.

Prisoner's Dilemma - Payoffs

An alternative expression of this situation is given in the following payoff matrix.

		Player B	
		Cooperate	Defect
Player A	Cooperate	R=3,R=3	S=0,T=5
	Defect	T=5,S=0	P=1,P=1

The payoffs are traditionally called:

- T – Temptation to defect
- R – Reward for mutual cooperation
- S – Sucker's Payoff
- P – Punishment for mutual defection

And the condition T>R>P>S must hold.

Prisoner's Dilemma - The Dilemma

To Cooperate or To Defect?

- If you think your opponent will cooperate, the rational choice is to defect to receive the higher payoff.
- If you think your opponent will defect, the rational choice is also to defect.

However your opponent will come to the same conclusion.

Thus the game, played with two rational players, will always result in mutual defection. This is unfortunate as both players could have scored higher if they had cooperated. The fact that rational logic can result in such a situation is the perplexing dilemma at the heart of this problem.

Iterated Prisoner's Dilemma

More interesting situations arise when we consider repeated plays of the Prisoner's Dilemma, the Iterated Prisoner's Dilemma.

The possibility of future interactions means that actions taken now could affect future payoffs, thus the simple 'defect always' conclusion no longer holds.

This allows players to develop more sophisticated strategies for game play which may take into account an opponents previous moves.

This version of the game was used in this project.

Why study the Prisoner's Dilemma?

This game may seem simple but it has generated a huge amount of research and has been used to analyze and explain a multitude of real world scenarios such as:

- businesses interacting in a market
- personal relationships
- super power negotiations
- trench warfare "live and let live" system of World War I

This project is predominantly concerned with applying the Prisoner's Dilemma to show how cooperation can evolve in a 'hostile' environment of selfish individuals. The Prisoner's Dilemma has proved a powerful tool for explaining the evolution of cooperation from Robert Axelrod's pioneering work to Richard Dawkin's use of it in his famous work "The Selfish Gene".

Genetic Algorithms - Concept

GA's use evolution as a search strategy.

Mimics evolution in the natural world:

- Natural Selection

  Darwinian 'Survival of the fittest'

- Natural Genetic operations

  Genetic operations of sexual reproduction such as crossover and mutation

Genetic Algorithms - Problems

- Representation
- Fitness Function
- Selection
- Reproduction
- Replacement

Genetic Algorithms - Representation

Genotype – the Prisoner's 'DNA'

A concise representation of the prisoner's strategy for playing the IPD.

Genotype encoded as a binary string, each bit representing the move to make (1 –C, 0 –D) based on the game history.

Each prisoner has a 3-game memory

4 possible results for a game (CC, CD, DC, DD)

4^3 = 64 bits, plus 7 to encode start game moves = 71bit string

Genetic Algorithms – Fitness Function

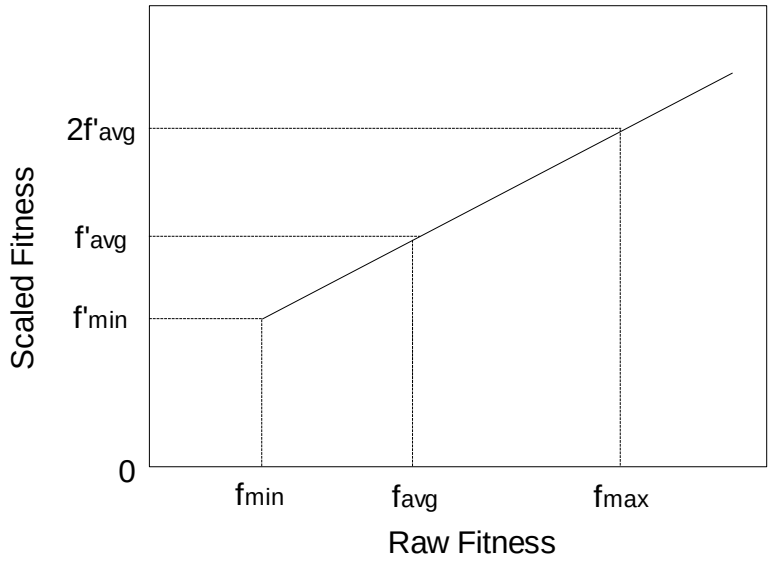
To evaluate how well a strategy is performing

Prisoner's dilemma has a natural fitness function, the game payoffs.

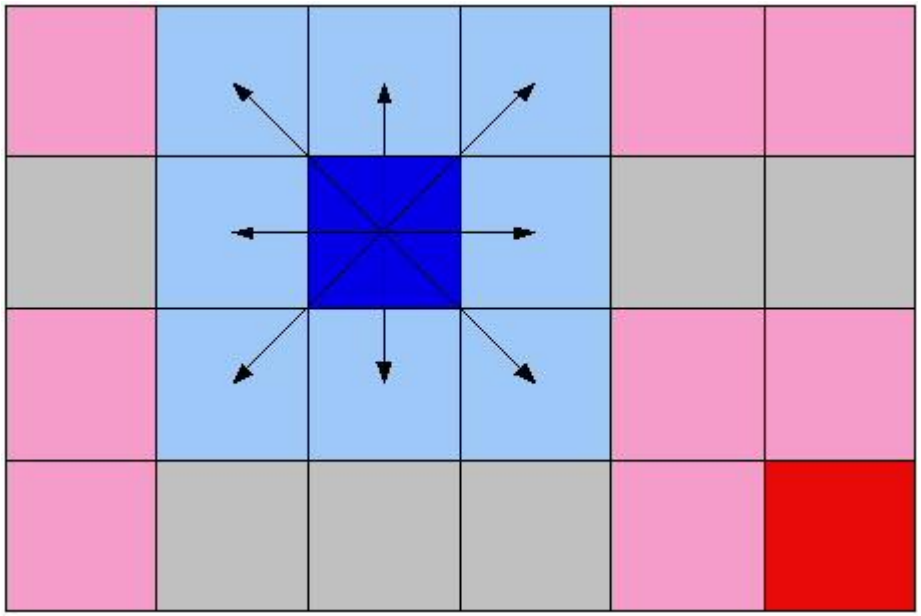
Two models:

- Tournament
- Spatial

Linear fitness scaling was performed.



Spatial Interactions



- **8 surrounding neighbours**
- **Overlapping edges**

Genetic Algorithms – Selection

Which Prisoner's should be allowed to reproduce?

Roulette-Wheel Selection

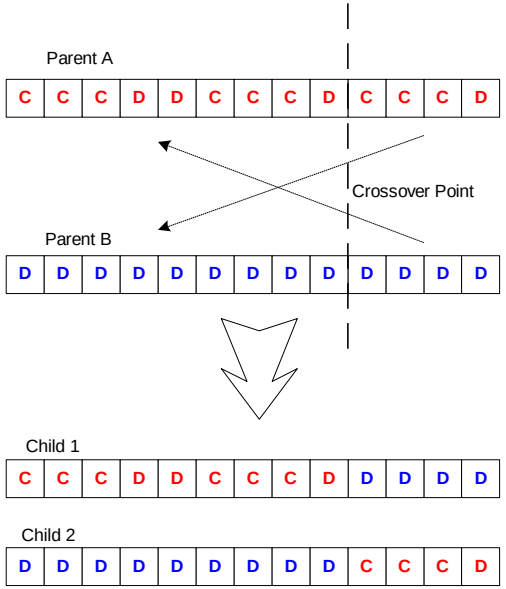
- Random spin of a roulette-wheel.
- Each slot represents a Prisoner.
- Probability of landing in each slot is weighted by the Prisoner's fitness.

Note: This (importantly) allows weak strategies to reproduce as well as the fittest.

Genetic Algorithms – Reproduction

Having selected two Prisoner's how can they produce offspring?

Crossover:



Mutation

With a (very low) probability flip a bit being copied from parent to child.

Genetic Algorithms – Replacement

How should the resulting offspring be added back to the population?

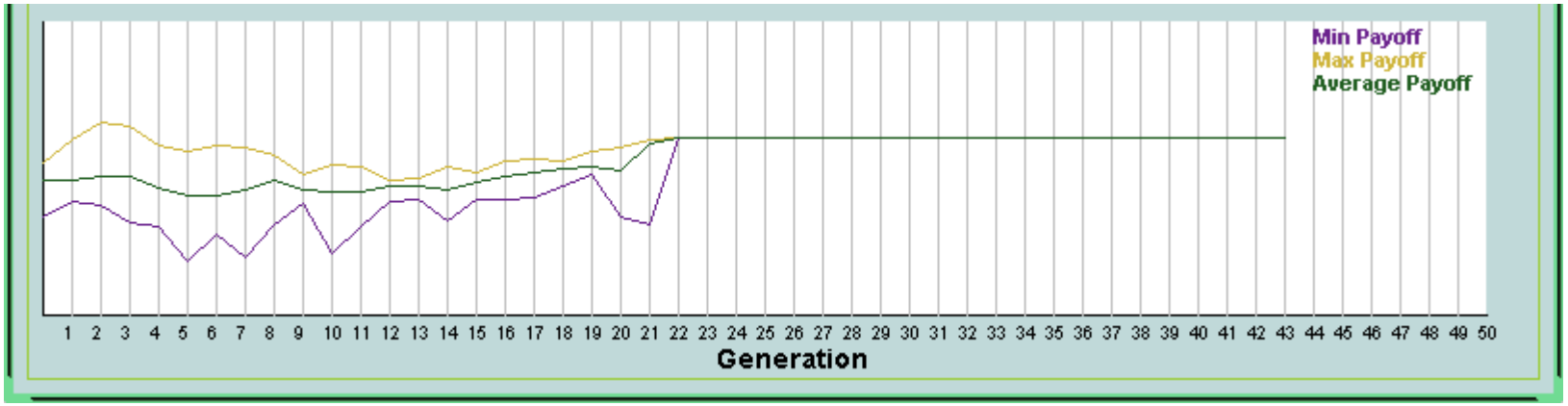
In *Tournament* mode the offspring go on to form a completely new population (non-overlapping).

In *Spatial* mode the offspring replace the weakest prisoner neighbouring the parent (overlapping).

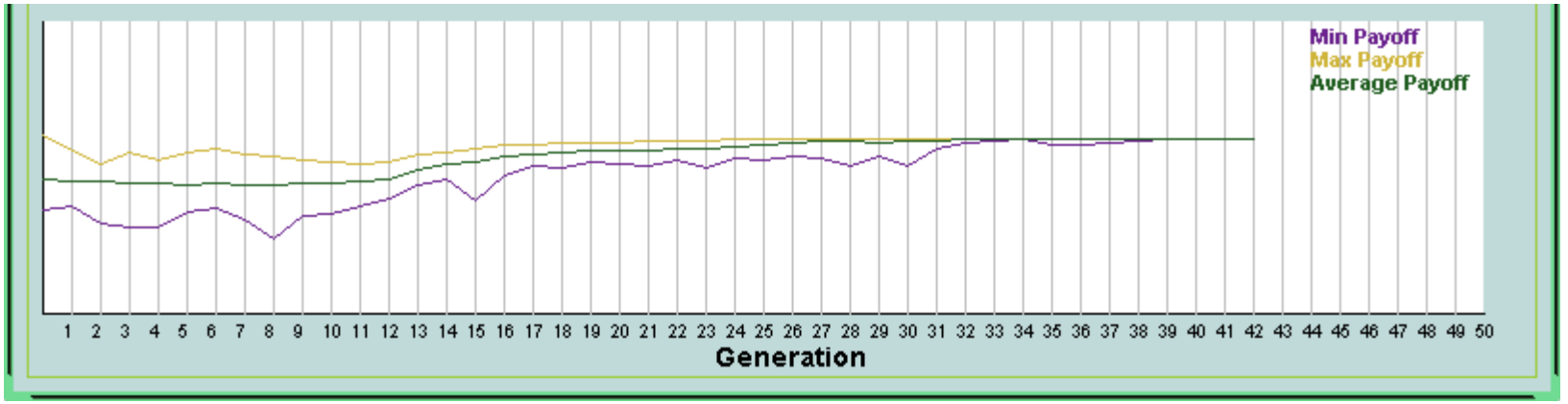
The Program

Tournament Results

Tournament



Population: 30, Iterations: 100

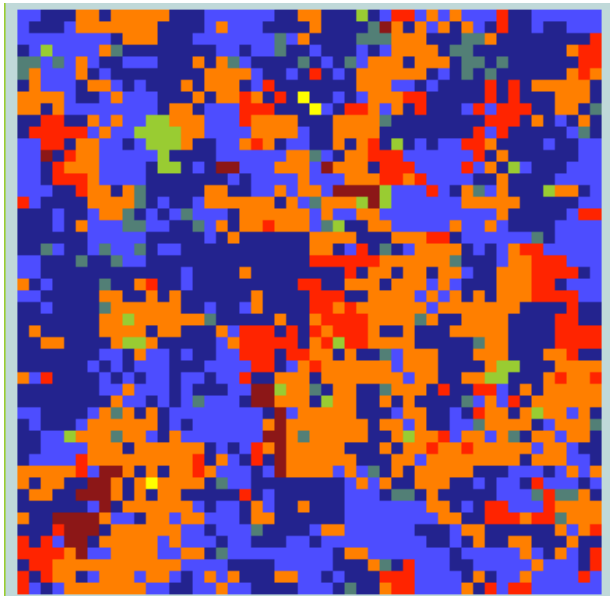


Population: 70, Iterations: 100

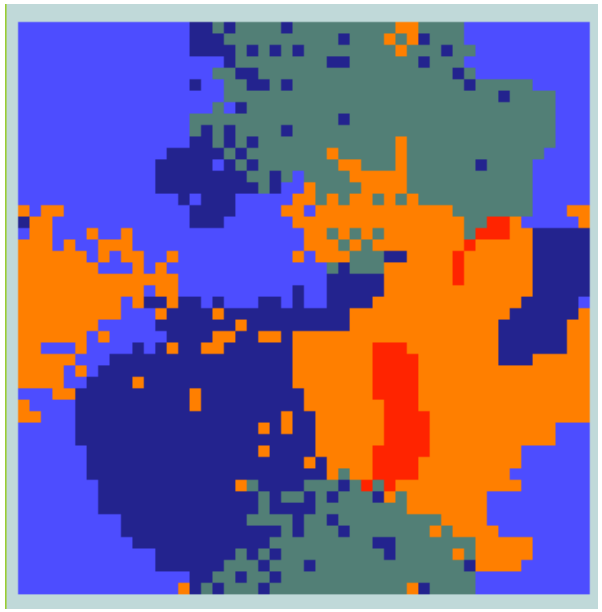
Spatial EA Results

Spatial - Evolutionary Algorithm

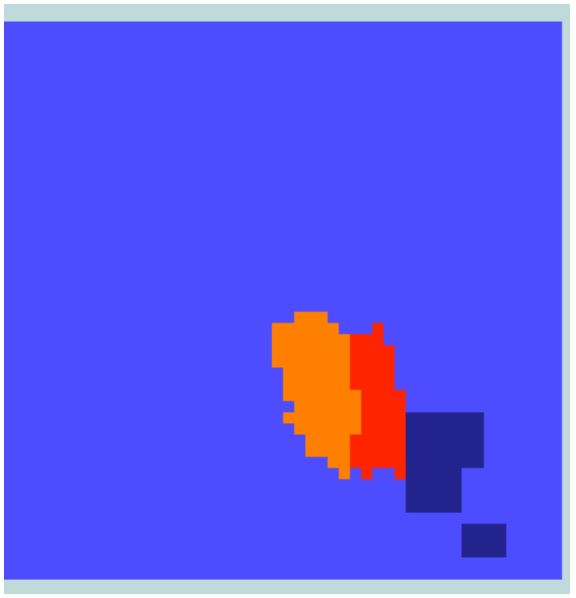
Population: 1225, Iterations: 100



Generation: 20



Generation: 75

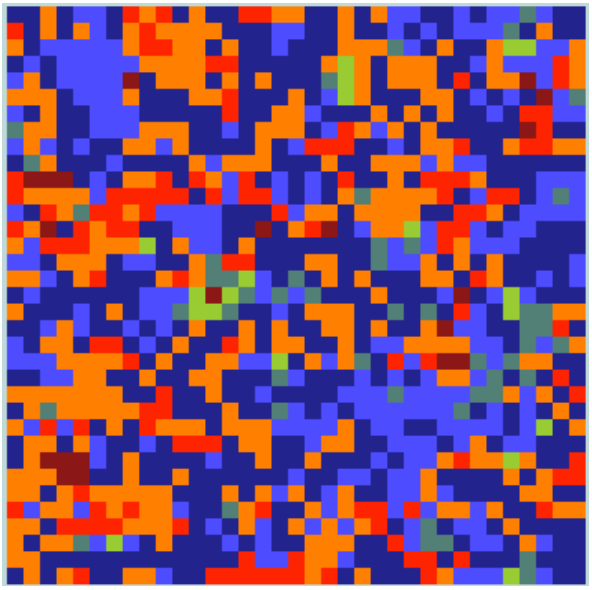


Generation: 150

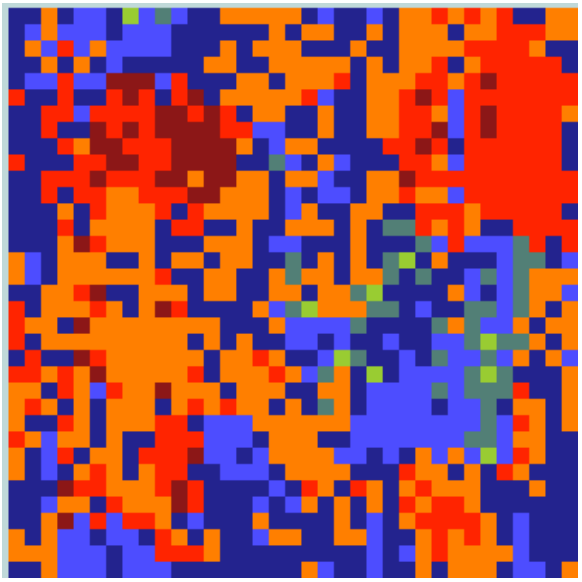
Spatial GA Results

Spatial - Genetic Algorithm

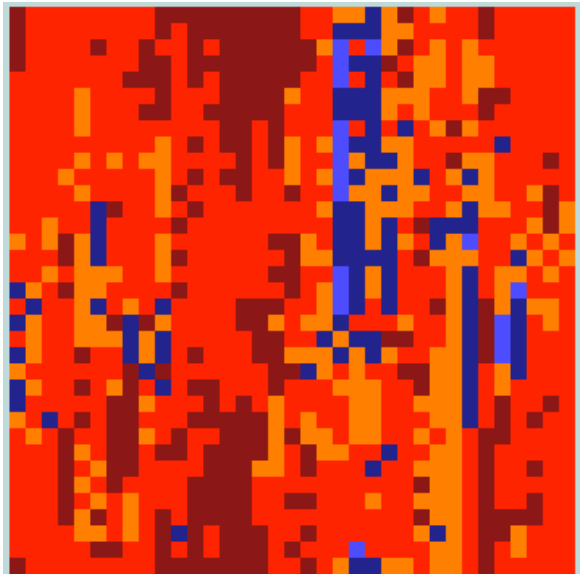
Population: 1225, Iterations: 100



Generation: 10



Generation: 75



Generation: 150

Problems Encountered

- Responsive GUI – difficulties in providing a responsive GUI while the genetic algorithm was running, solved using Multi-threading.
- Premature convergence – this occurred regularly in early versions. Fitness scaling helped prevent this.
- Out of Memory errors – When running large simulations, while HW plays a part efforts were made to improve program efficiency (e.g. modifying some data structures to Hash tables)
- Speed – Program was initially very slow, this was overcome by using faster data structures and improving the efficiency of some code segments.

Possible Improvements

- Saving - Saved Strategies and Rule settings
- Genetic Algorithm – alternative selection, fitness scaling and replacement techniques
- Seeded initial populations
- Strategy Creator – allow user to create their own custom strategies
- Improved Randomness – using a CSPRNG
- GUI improvements – better user support
- Prisoner Analysis – allow user to click on a prisoner in population and view their stats

Questions

?