Problem solving exercise class N-player Iterated Prisoner's Dilemma (co-evolution case study)

Ata Kaban

University of Birmingham

Before we start

Your comments on feedback forms:

- “need more exercise classes”

We will do exercise class / case study sessions from time to time.

But, have you:

- worked through the “Mid-term Practice Quiz” released last week on Canvas?
- tried to solve past exam questions?

Before we start

Your comments on feedback forms:

- Assignment question on Exercise Sheet 1 was "poorly formulated"..."badly explained"

Hmm... Note: This q was from past exam papers!

Advise: Use my office hours to ask questions when you need clarification. (Not the day before a deadline or a day before exam.)

Iterated Prisoner's Dilemma

- Invented by Merrill Flood & Melvin Dresher in 1950s
- Studied in game theory, economics, political science
- The story
  - Alice and Bob arrested, no communication between them
  - They are offered a deal:
    - If any of them confesses & testifies against the other then gets suspended sentence while the other gets 5 years in prison
    - If both confess & testify against the other, they both get 4 years
    - If none of them confesses then they both get 2 years
  - What is the best strategy for maximising one’s own payoff?

N-Player Version

The payoff matrix of the N-player iterated prisoner’s dilemma game, for Player A is:

	0	1	2	...	N-1
Coop	0	2	4		2(N-1)
Defect	1	3	5		2(N-1)+1

All players are treated equally.

Design a co-evolutionary algorithm for learning to play the iterated 4-player prisoner’s dilemma game.

- Chromosome representation for strategies (players)
- Fitness evaluation function
- Evolutionary operators (crossover, mutations)
- Selection scheme
- Comment on parameters of your design.
- Comment on strengths and weaknesses of your design

Recap: 2-player version of the game (To skip the recap, jump to slide 13)

- Abstract formulation through a payoff matrix

		Player A	
		Cooperate	Defect
Player B	Cooperate	3,3	0,5
	Defect	5,0	1,1

- 2 tournaments – participants have sent strategies
- Human strategies played against each other
- Winner: TIT FOR TAT
  - Cooperates as long the other player does not defect
  - Defects on defection until the other player begins to cooperate again

- Can GA evolve a better strategy?

- Individuals = strategies
- How to encode a strategy by a string?

- Let memory depth of previous moves=1

Fix a canonical order of cases:

	A	B
– Case 1:	C	C
– Case 2:	C	D
– Case 3:	D	C
– Case 4:	D	D

e.g. strategy encoding (for A): ‘CDCD’

- Now let memory depth of previous moves=3
  - How many cases? ………
    - Case 1: ……..
    - Case 2: ……..
    - …
  - How many letters are needed to encode a strategy as a string? ……………
  - How many strategies there are? ………….
    - Is that a large number?

- Experiment 1
  - 40 runs with different random initialisations
  - 50 generations each
  - Population of 20
  - Fitness=avg score over all games played
  - A fixed environment of 8 human-designed strategies
- Results
  - Found better strategy than those 8 strategies in the environment!
  - Even though – how many strategies were only tested in a run out of all possible strategies? …………….
  - What does this result mean? …………….

- Experiment 2
  - changing environment: the evolving strategies played against each other.
- Results
  - Found strategies similar in essence with the winner human-designed strategy
- Idealised model of evolution & co-evolution

N-Player Version

The payoff matrix of the N-player iterated prisoner's dilemma game, for Player A is:

	0	1	2	...	N-1
Coop	0	2	4		2(N-1)
Defect	1	3	5		2(N-1)+1

All players are treated equally.

Representation

- Strategy = lookup table
  - Situation (history) → action, for each situation

- How to represent history of the game?
  - Let *l* denote the length of the history considered

- How many histories are possible in this game?

…representing history

- The player's own previous *l* moves
  - Requires ……….. bits
- The number of co-operators in the last *l* moves
  - Requires ……….. bits

→That is ………. bits in total

- An example of encoded history, if *l*=3:

  001111001

  What does it mean?

  ➢ need a convention as of which bit means what

    - Let the first *l* bits indicate the player's own actions
      - Let the leftmost bit refer to the most recent move
    - Let the next groups of 2 bits indicate the nos of collaborators
      - Let the leftmost group refer to the most recent move

  001 11 10 01

  Now the bit-string 'makes sense'!

  Can you read the story from the bit-string now?

- How many histories are there in total?

  If 9 bits are needed to represent a history

  Then there are 2^9 histories possible.

  Remember, we agreed that strategies will be stored as 'lookup tables':

  One strategy is a binary string (0=coop, 1=defect) that gives an action for all possible histories. How long this string is?……………

  So 2^9 bits are needed to represent a strategy.

  e.g. for history '001 11 10 01'=121, the action is whatever is listed in entry (bit) 121.

- Sure? Anything missing?

- What is missing?
  - Actions are taken as function of the history
  - What about the very first action?

- Need some extra bits to represent the “pre-history” of *l=3* virtual previous rounds at the beginning of the game!
  - That is 9 extra bits. (But there several possibilities here.)
- Length of bit string that represents one strategy:
  - It’s not 2^9 but 2^9+9
-

Q: Would you be able to write this quantity more generally, with history length l and nos of players N?

Fitness evaluation function

- Fitness of an individual player is evaluated by playing a number K of (N-player) games with adversaries randomly drawn from the population.
- Adding the payoffs obtained by each individual is a measure of its fitness.

Evolutionary operators

- Since binary strings are used, standard operators are available, e.g.
  – Uniform crossover
  – Bit-wise flipping can be used

Selection scheme

- Fitness ranking or tournament selection
- Important is to maintain the *selection pressure* constantly!

Discussion. Strengths, weaknesses

- Strengths:
  - Generic, the same design can be applied for more general number of players N
  - Simplicity in evolution and game playing due to bit string representation
- Weaknesses:
  - In N is large, computation time is long
  - Inability to capture multiple cooperation levels
- Parameters that influence the results:
  - History length
  - Nos of generations