ETH zürich

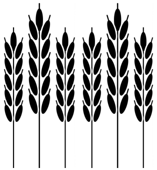
Evolution of Strategies for Prisoner's Dilemma

Gleb Fedorovich, Jan Heldmann, Lukas Schmitt, Fengshi Zheng

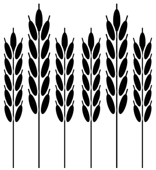
Historical background

Bob

Today



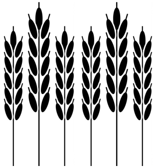
Tomorrow



Alice

Historical background

Tomorrow

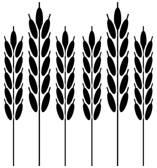


Alice

Bob

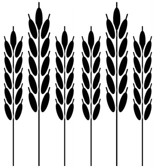
	A Cooperates	A Defects
B Cooperates	(3,3)	(0,5)
B Defects	(5,0)	(1,1)

Today



Historical background

Tomorrow

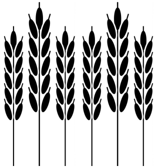


Alice

Bob

	A Cooperates	A Defects
B Cooperates	(3,3)	(0,5)
B Defects	(5,0)	(1,1)

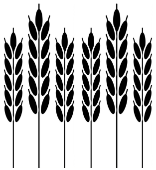
Today



Historical background

Alice

Tomorrow

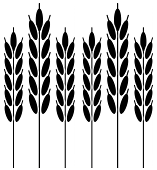


Bob

	A Cooperates	A Defects
B Cooperates	(3,3)	(0,5)
B Defects	(5,0)	(1,1)

Nash Equilibrium

Today



Axelrod’s tournament

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

Axelrod's tournament

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

TFT											
rand.											

Axelrod's tournament

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

TFT	C										
rand.	C										

Axelrod’s tournament

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

TFT	C	C									
rand.	C	D									

Axelrod's tournament

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

TFT	C	C	D								
rand.	C	D	D								

Axelrod’s tournament

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

TFT	C	C	D	D	D	C	C	D	C	C	
rand.	C	D	D	D	C	C	D	C	C	D	

Axelrod's tournament

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

TFT	C	C	D	D	D	C	C	D	C	C	Total = 21
rand.	C	D	D	D	C	C	D	C	C	D	Total = 26

Axelrod’s tournament

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

TFT	C	C	D	D	D	C	C	D	C	C	Total = 21
rand.	C	D	D	D	C	C	D	C	C	D	Total = 26

Important:

- TFT can never get more points than the opponent
- TFT is not the best but the most robust strategy

Axelrod's tournament

	A (C)	A (D)
B (C)	(3,3)	(0,5)
B (D)	(5,0)	(1,1)

- Participants played the Iterated Prisoner's Dilemma (IPD)
- Tit For Tat (TFT) was the winning Strategy

TFT	C	C	D	D	D	C	C	D	C	C	Total = 21
rand.	C	D	D	D	C	C	D	C	C	D	Total = 26

Important:
- TFT can never get more points than the opponent
- TFT is not the best but the most robust strategy

4 Reasons TFT won:
- 1. Starts with cooperation which is nice and prevents trouble
- 2. Punishes the defection of its opponent which stimulates cooperation
- 3. Forgives the opponent when they cooperate again
- 4. TFT is a clear strategy which is easy for an opponent to anticipate

Description of Agents

The agents implemented in our model have various types of strategies

Description of Agents

The agents implemented in our model have various types of strategies:

- Always cooperating
- Always defecting

Description of Agents

The agents implemented in our model have various types of strategies:

- Always cooperating
- Always defecting
- Tit-for-tat (**TFT**): if you cooperate, I'll cooperate. Otherwise I defect

Description of Agents

The agents implemented in our model have various types of strategies:

- Always cooperating
- Always defecting
- Tit-for-tat (**TFT**): if you cooperate, I'll cooperate. Otherwise I defect
- Neural-Agents: cooperate or defect depends on the result of a neural network

Description of Agents

The agents implemented in our model have various types of strategies:

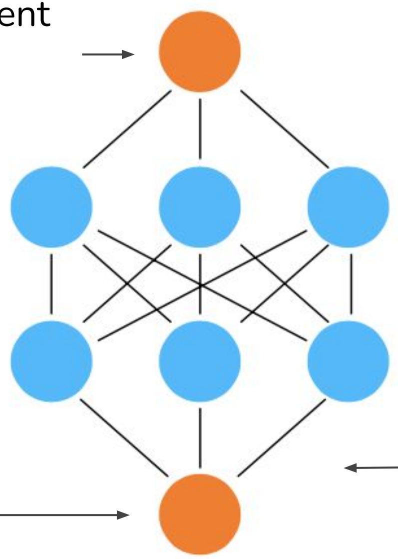
- Always cooperating
- Always defecting
- Tit-for-tat (**TFT**): if you cooperate, I'll cooperate. Otherwise I defect
- Neural-Agents: cooperate or defect depends on the result of a neural network
- String-Agents: follow a set of deterministic rules of action

Neural-Agents

Take the history(ies) of the last tournament (0 - cooperating, 1 - defecting)

Feed the action through a forward network

Take the output of the network (float point number from 0 to 1) as the tendency to defect (stochastic)



Each layer applies a linear transform of the previous layer's output, followed by a non-linear activation function:

$$\boldsymbol{y}_{out} = \boldsymbol{\sigma}\left(\boldsymbol{W} \cdot \boldsymbol{x} + \boldsymbol{b}\right)$$

Sigmoid function applied, to make the output between 0 and 1

String-Agents

Histories	Decision
(CC), (CC)	C
(CC), (CD)	D
(CC), (DC)	C
(CC), (DD)	D
…	…
(DD), (DD)	D

- A lookup table that tells the agent which action to take;
- Histories: a list of (my action, opponent's action)

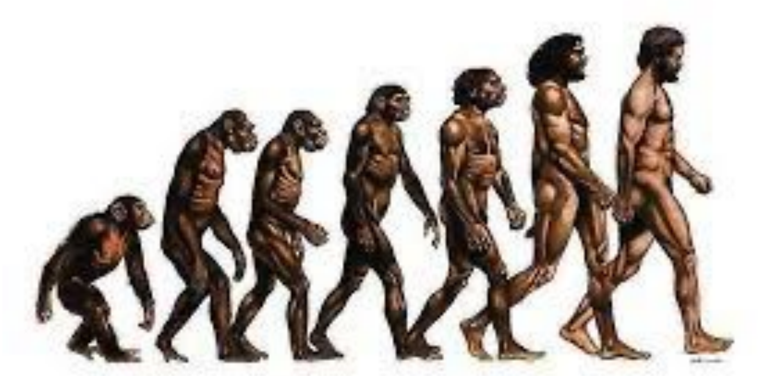
String-Agents

Histories	Decision
(CC), (CC)	C
(CC), (CD)	D
(CC), (DC)	C
(CC), (DD)	D
…	…
(DD), (DD)	D

- A lookup table that tells the agent which action to take;
- Histories: a list of (my action, opponent's action)
- Take all the decisions in the lookup table, we obtain a **string-representation** for the agent: CDCD…D (TFT)

Genetic Algorithm

- An algorithm to find the maxima of a system
- Inspired by the evolution in nature



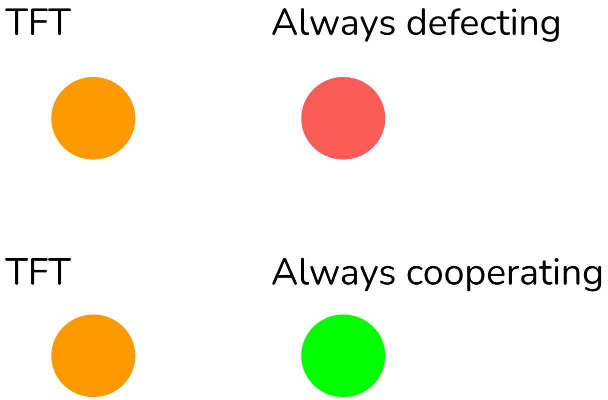
Always cooperating

Always defecting

TFT

Genetic Algorithm

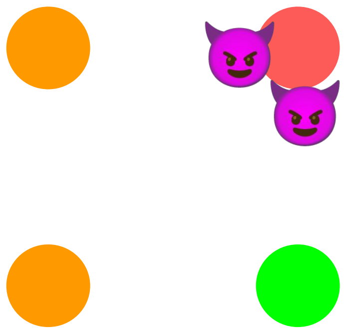
- Agents are born with different genes, which determines their behavior totally
  - E.g. String-Agents' string
  - Neural-Agents' weights and biases



Genetic Algorithm

- Agents are born with different genes
- In each generation, the agents play against neighbors. Their **payoffs** determine their "fitness in the nature"

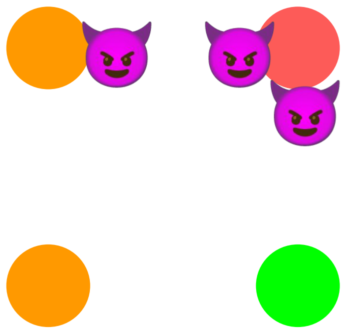
First round



Genetic Algorithm

- Agents are born with different genes
- In each generation, the agents play against neighbors. Their **payoffs** determine their "fitness in the nature"

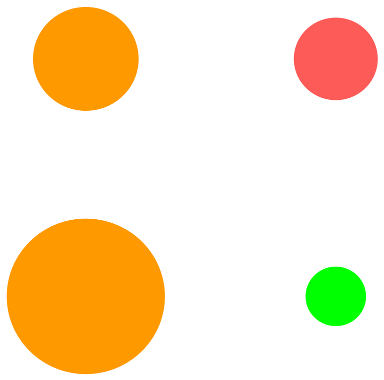
Subsequent rounds



Genetic Algorithm

After last round of the generation

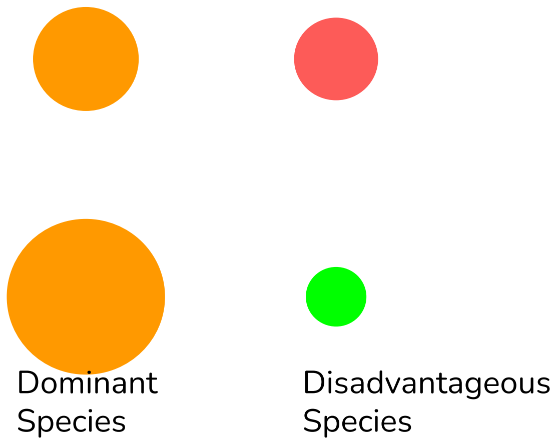
- Agents are born with different genes
- In each generation, the agents play against neighbors. Their **payoffs** determine their "fitness in the nature"



Genetic Algorithm

- Agents are born with different genes
- In each generation, the agents play against neighbors. Their **payoffs** determine their "fitness in the nature"
- The fitter agents will survive, whereas the weaker ones die

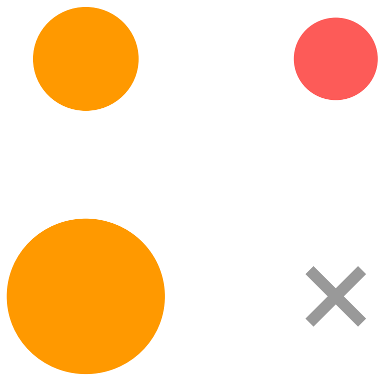
After last round of the generation



Genetic Algorithm

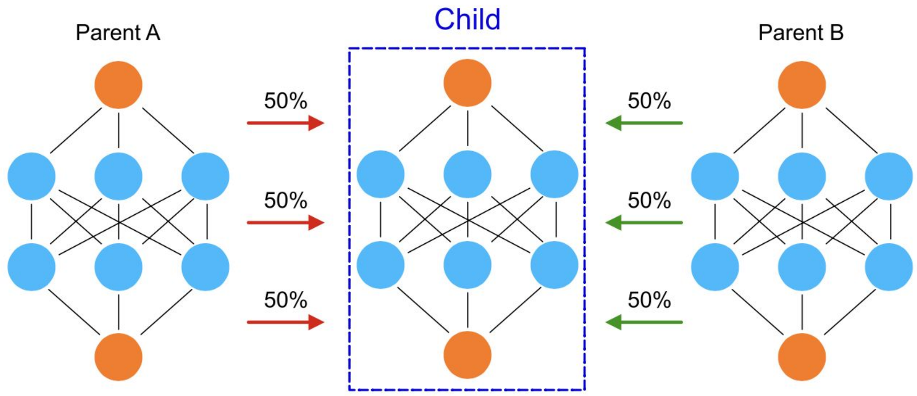
After last round of the generation

- The fitter agents will survive, whereas the weaker ones die
- The fittest agents reproduce with each other



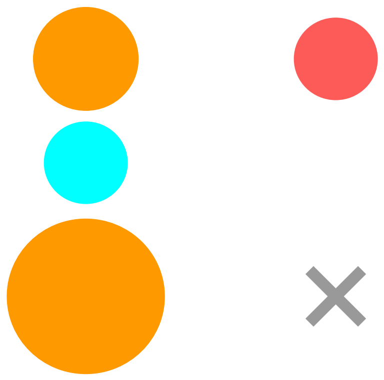
Genetic Algorithm

- The fitter agents will survive, whereas the weaker ones die
- The fittest agents reproduce with each other
    - Parents cross over their genes (or swap weight matrices) to create the children



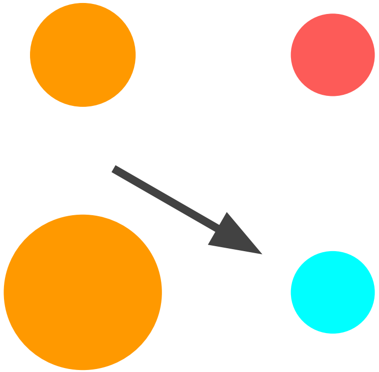
Genetic Algorithm

- The fittest agents reproduce with each other
  - Parents cross over their genes (or swap weight matrices) to create the children
  - Random mutations can happen to children's genes



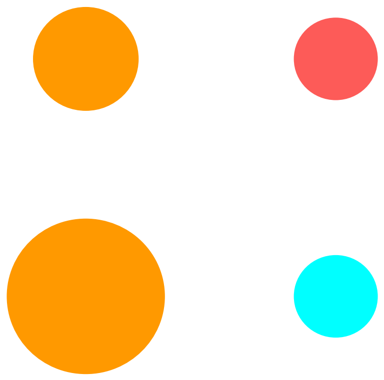
Genetic Algorithm

- The fittest agents reproduce with each other
  - Parents cross over their genes (or swap weight matrices) to create the children
  - Random mutations can happen to children’s genes
  - The child replace the weakest neighbor

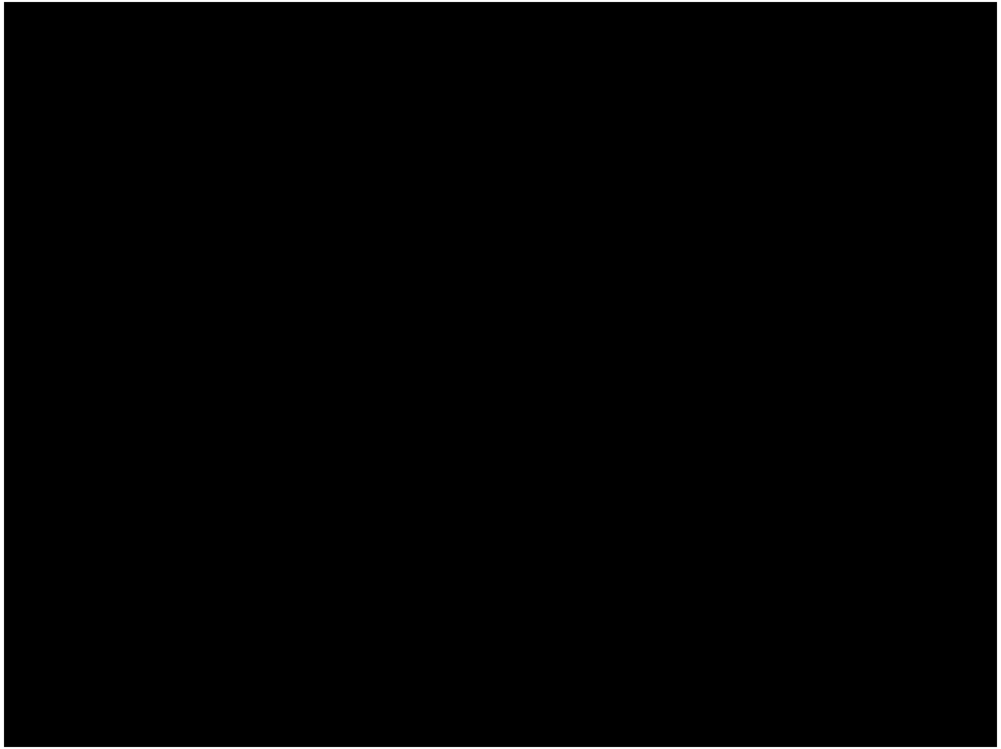


Genetic Algorithm

- The fittest agents reproduce with each other
    - Parents cross over their genes (or swap weight matrices) to create the children
    - Random mutations can happen to children’s genes
    - The child replace the weakest neighbor
- Redo the former process for next generations



Genetic Algorithm



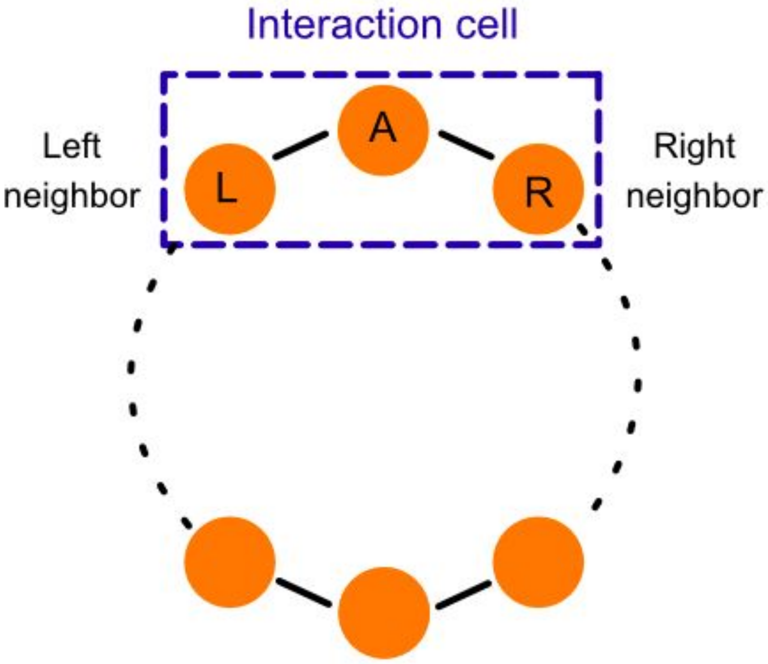
Project goals

Can the Agents learn Tit for Tat

Investigating the behavior of Agents in different settings

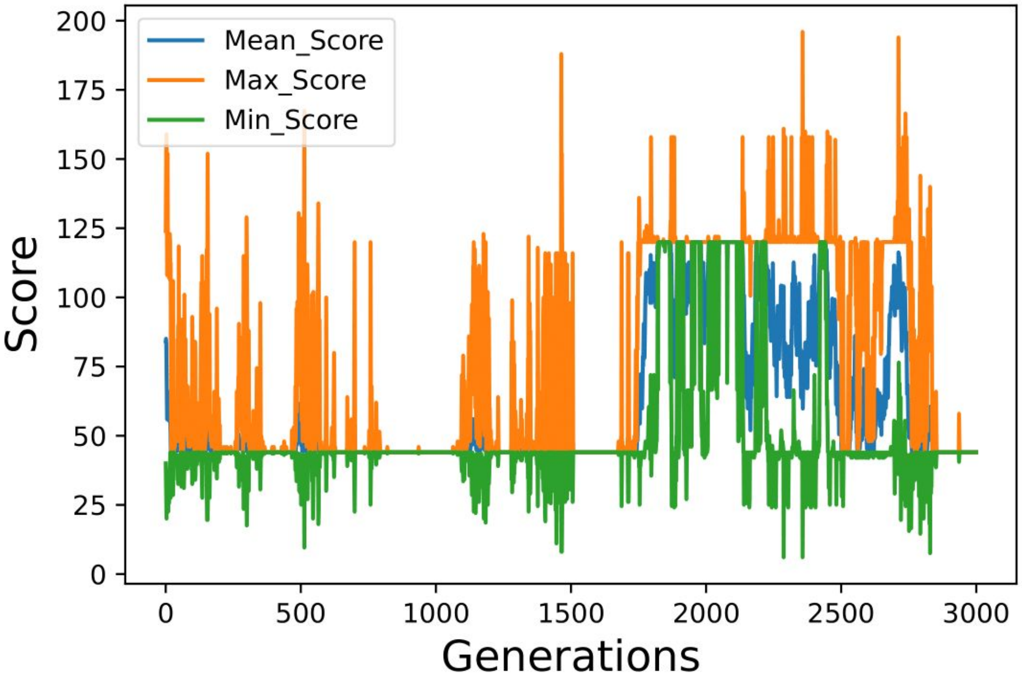
Analyzing the formation of cooperation

Ring Structure



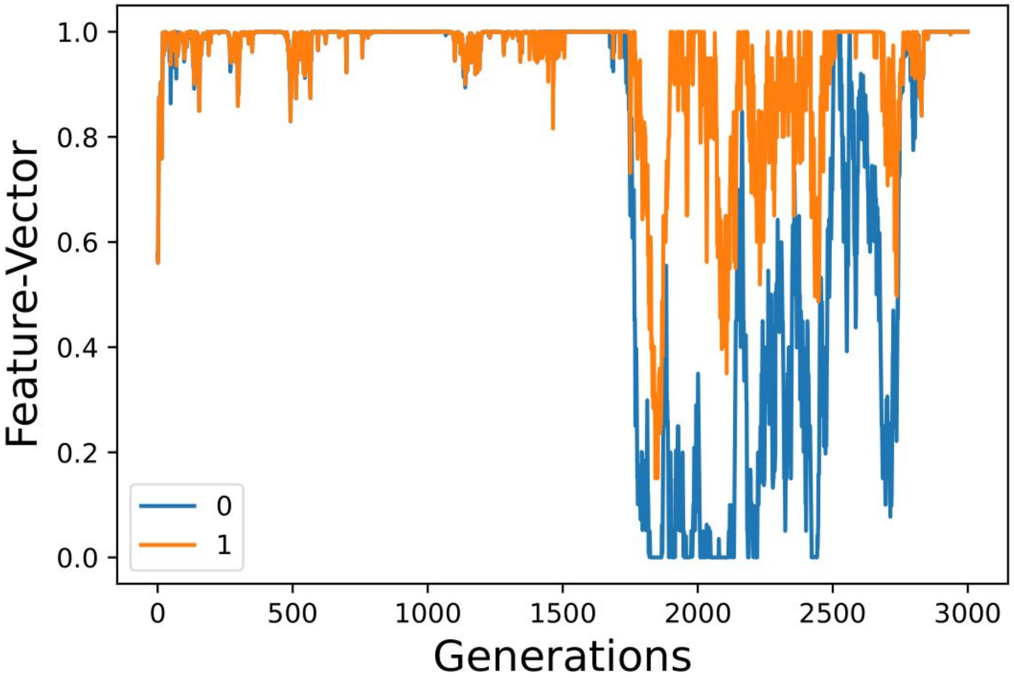
Ring Structure

- 20 Agents
- Nearest-Neighbor-Interaction
- 20 Rounds of PD
- Global-Genetic-Algorithm



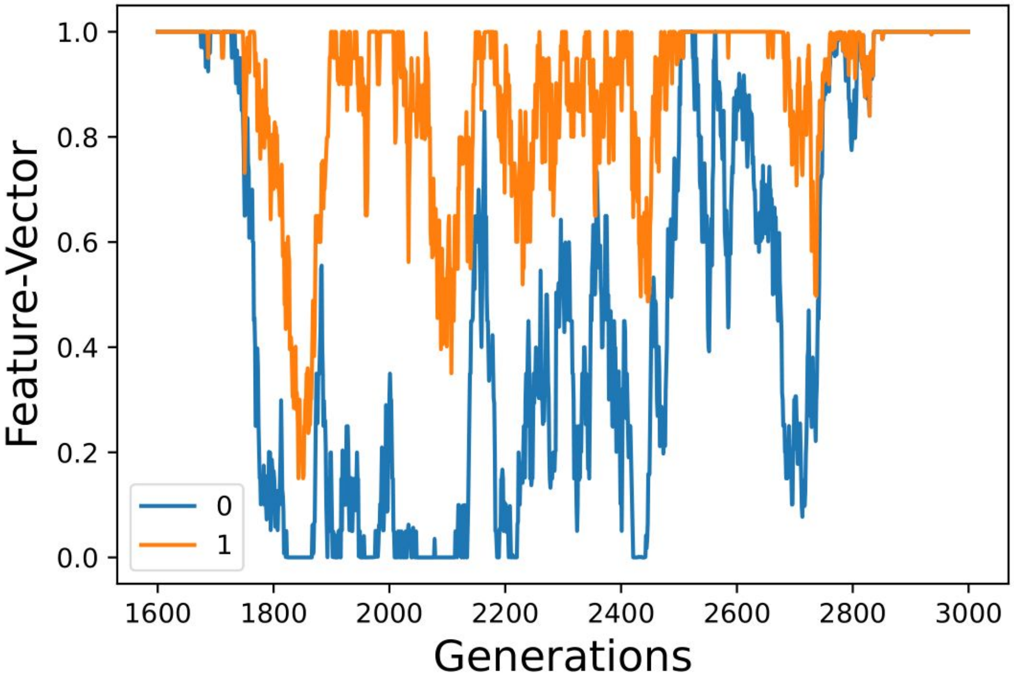
Ring Structure - Feature Vectors

- **FV0**: Output, given C of opponent
  - FV0 = 0: Cooperating
  - FV0 = 1: Defecting
- **FV1**: Output, given D of opponent
  - FV1 = 0: Forgiving
  - FV1 = 1: Punishing
- Phase of cooperation from 1700 to 2500



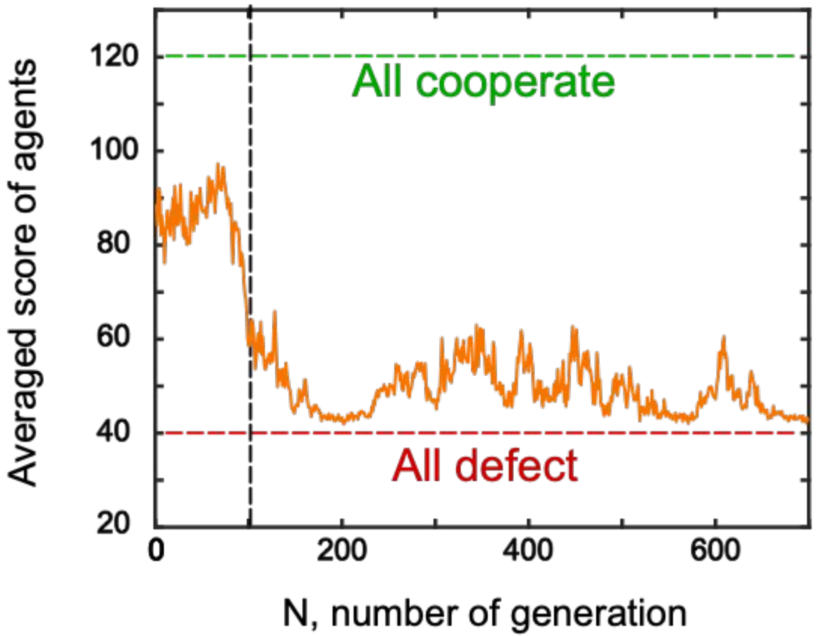
Ring Structure - Feature Vectors

- Starts with Tit for tat
- Always Cooperating
- Defectors spread
- Tit for Tat again



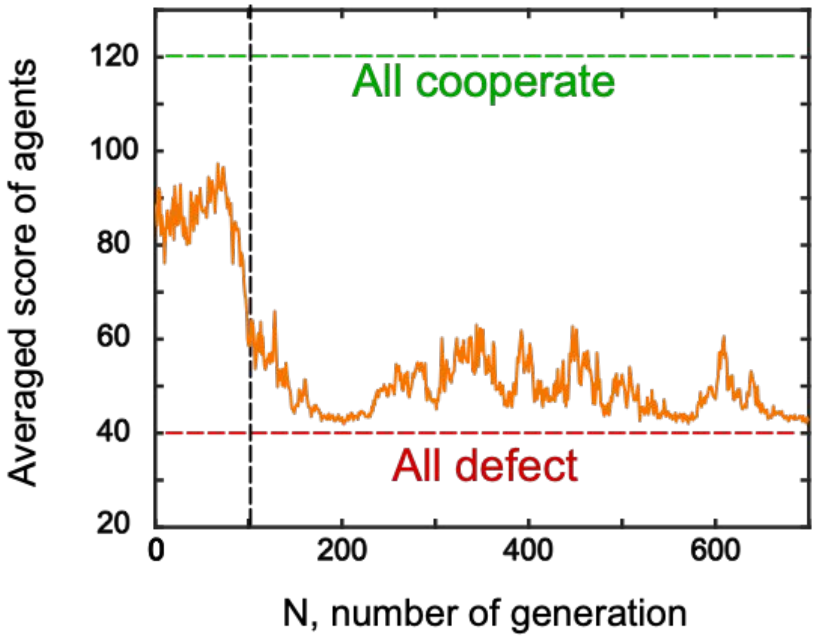
Ring - Influence from outside

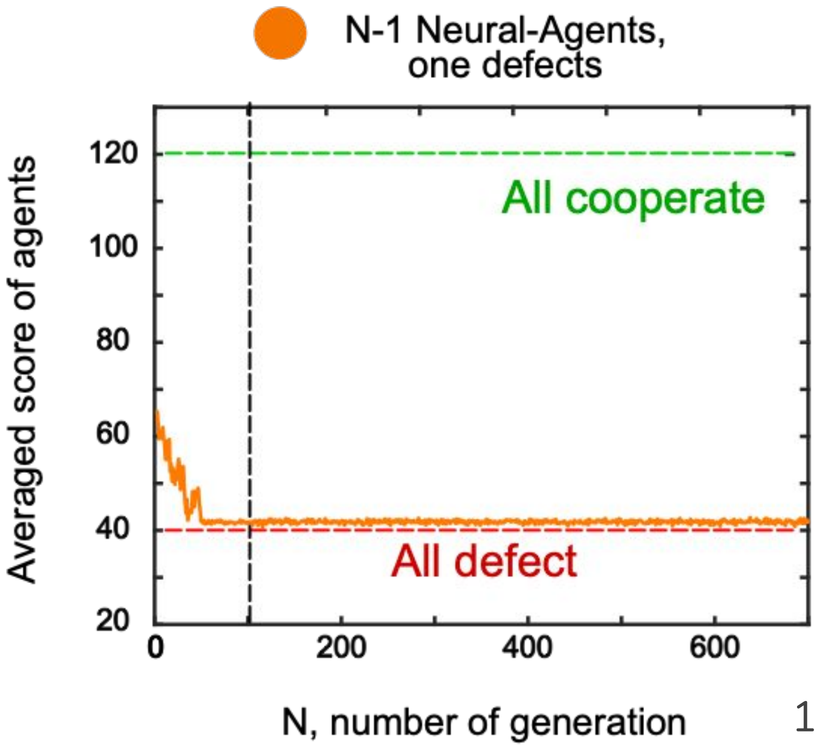
What we already know
For Neural-Agents?



Ring - Influence from outside

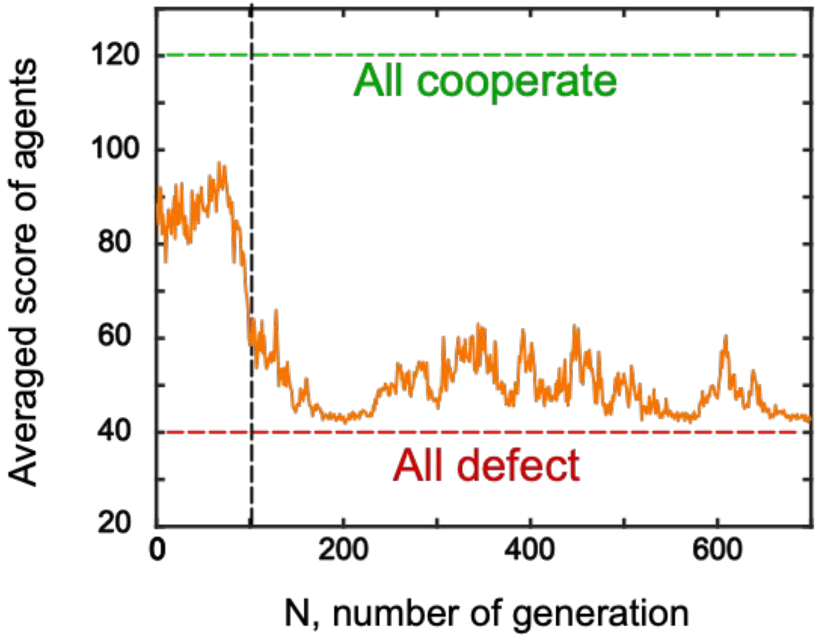
What we already know
For Neural-Agents?

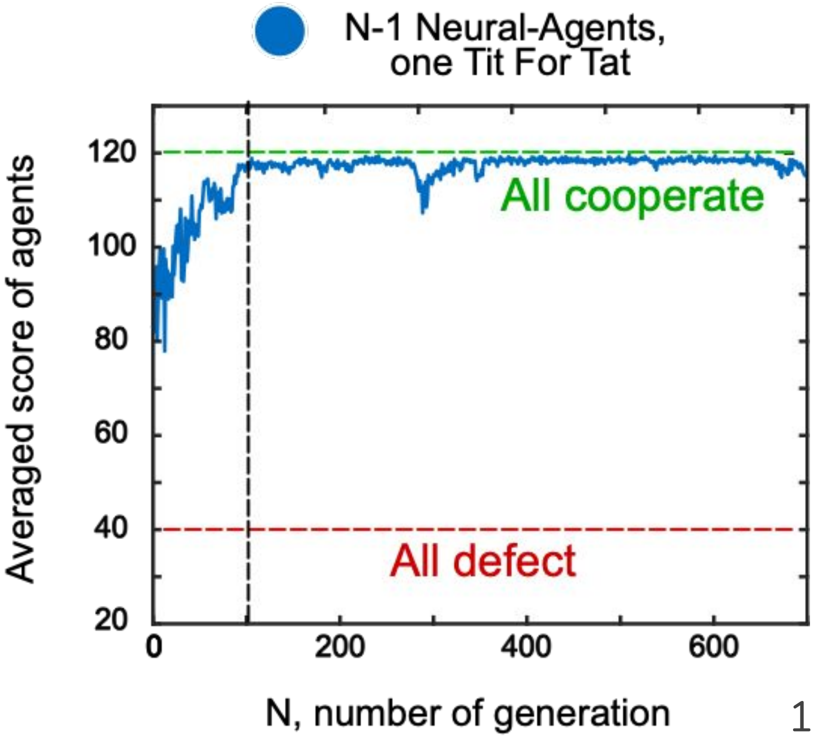




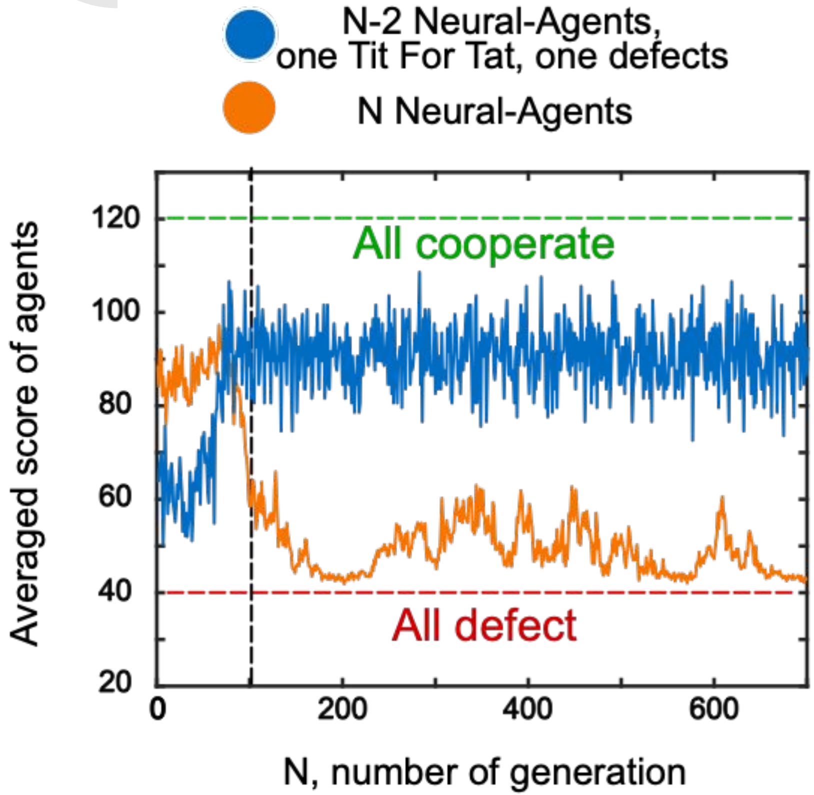
Ring - Influence from outside

What we already know
For Neural-Agents?

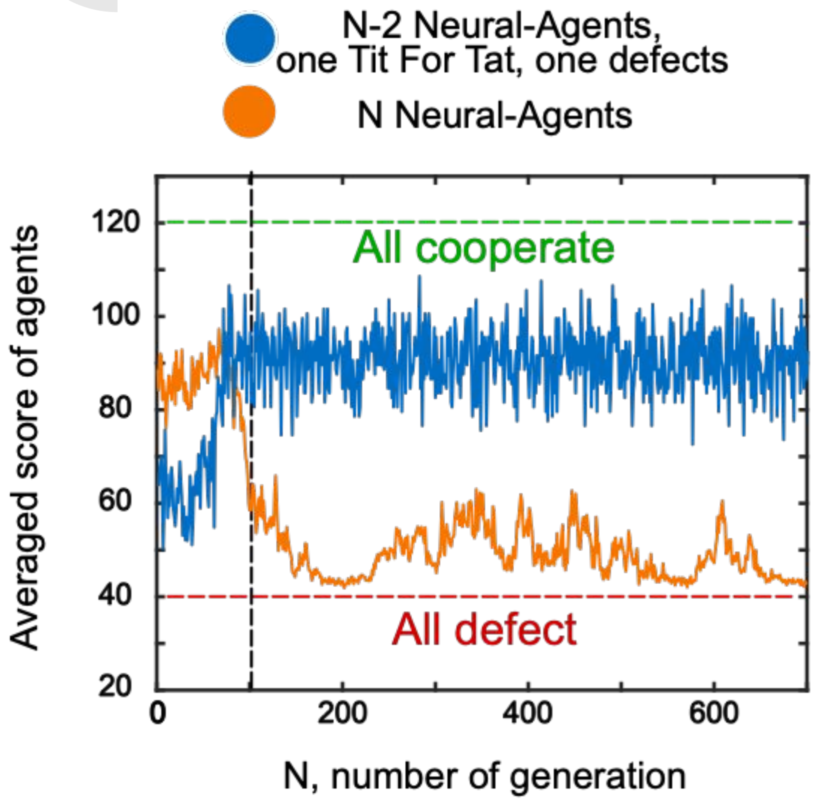


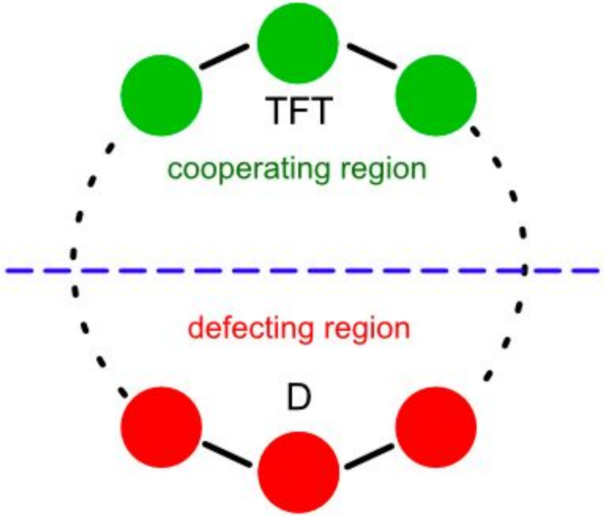


Ring - Influence from outside

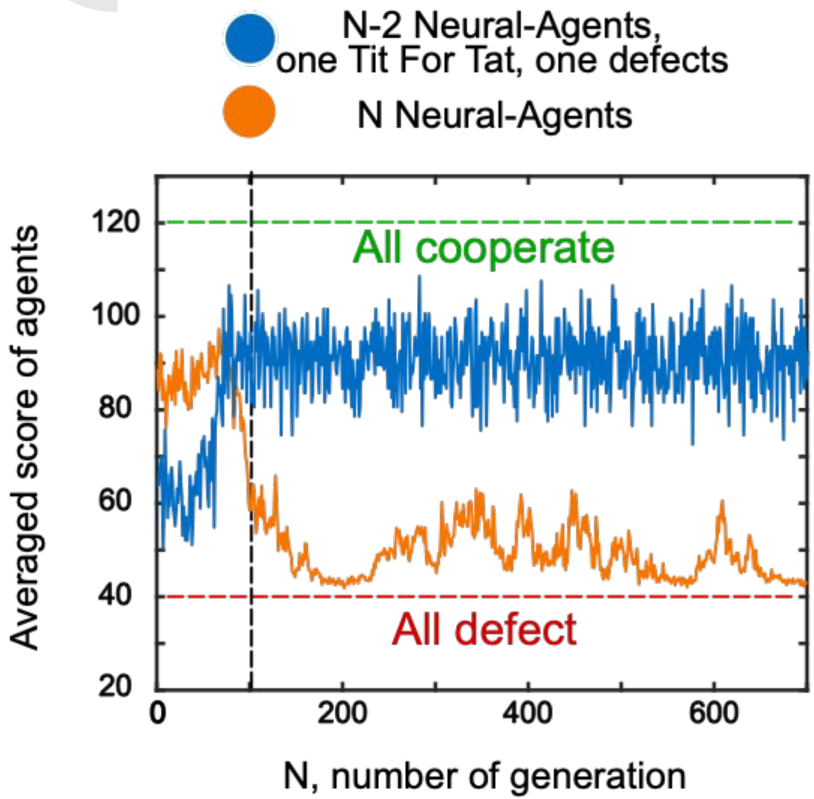


Ring - Influence from outside

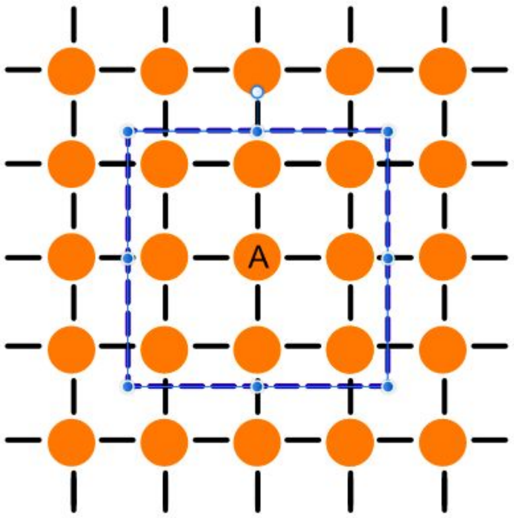




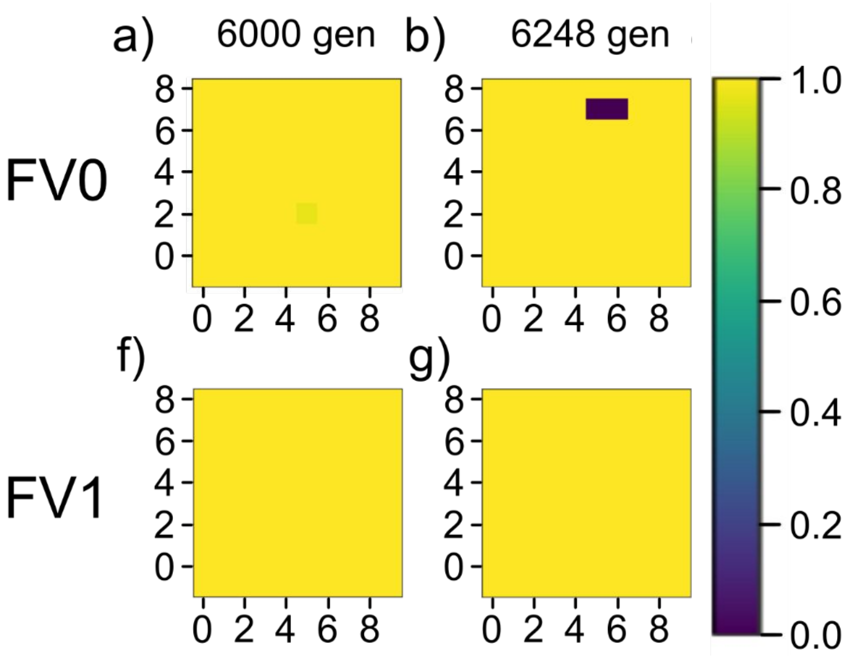
Ring - Influence from outside



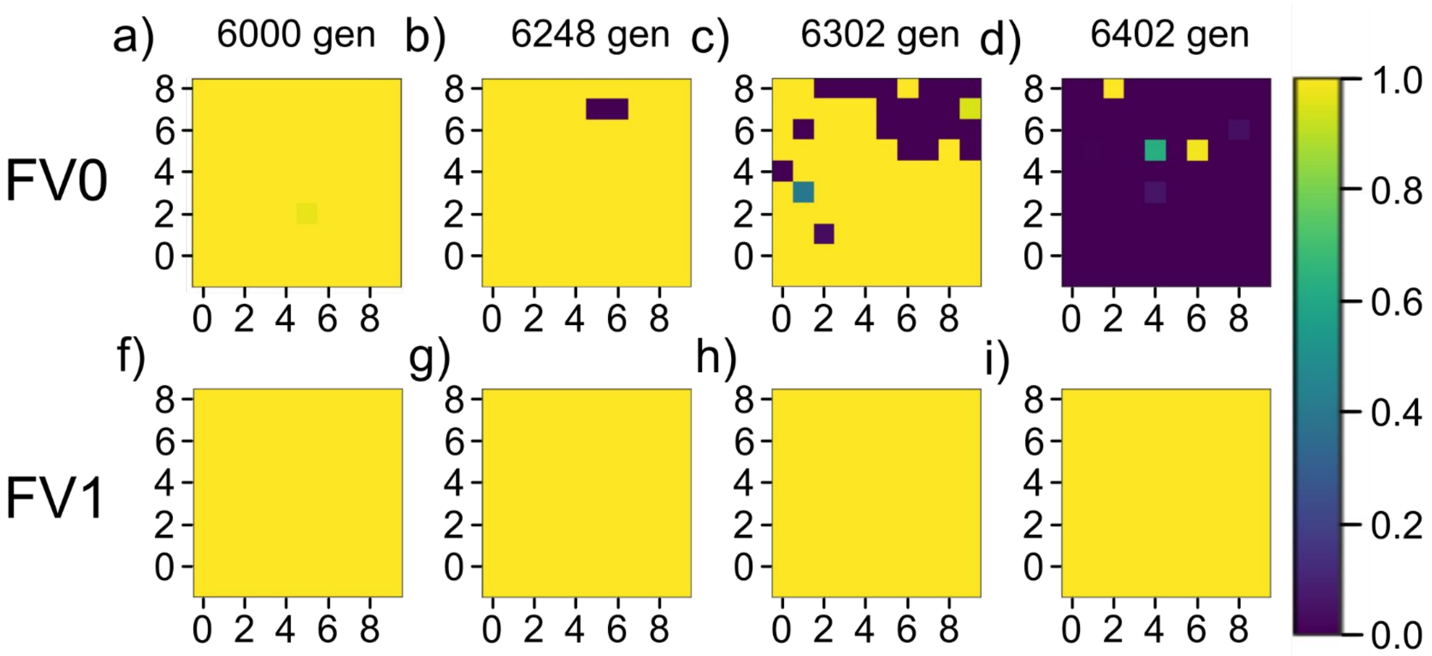
! Let’s go to 2D !



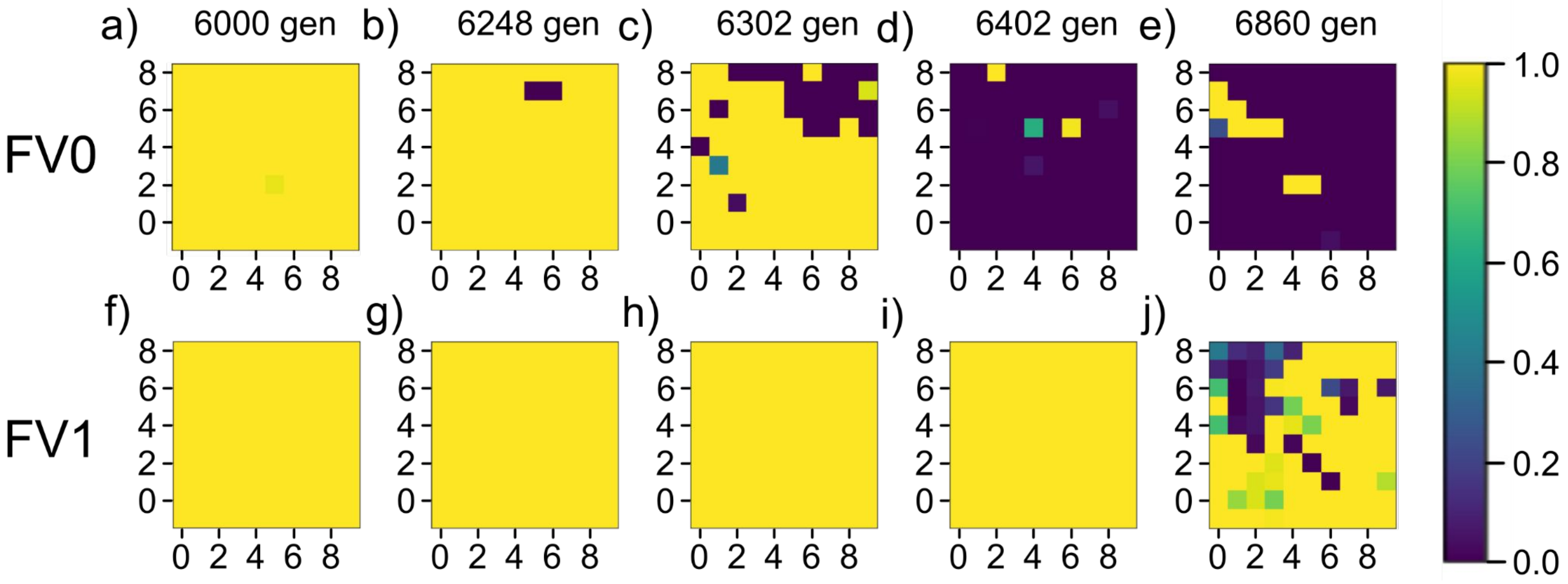
Grid - Formation of Cooperation



Grid - Formation of Cooperation



Grid - Formation of Cooperation

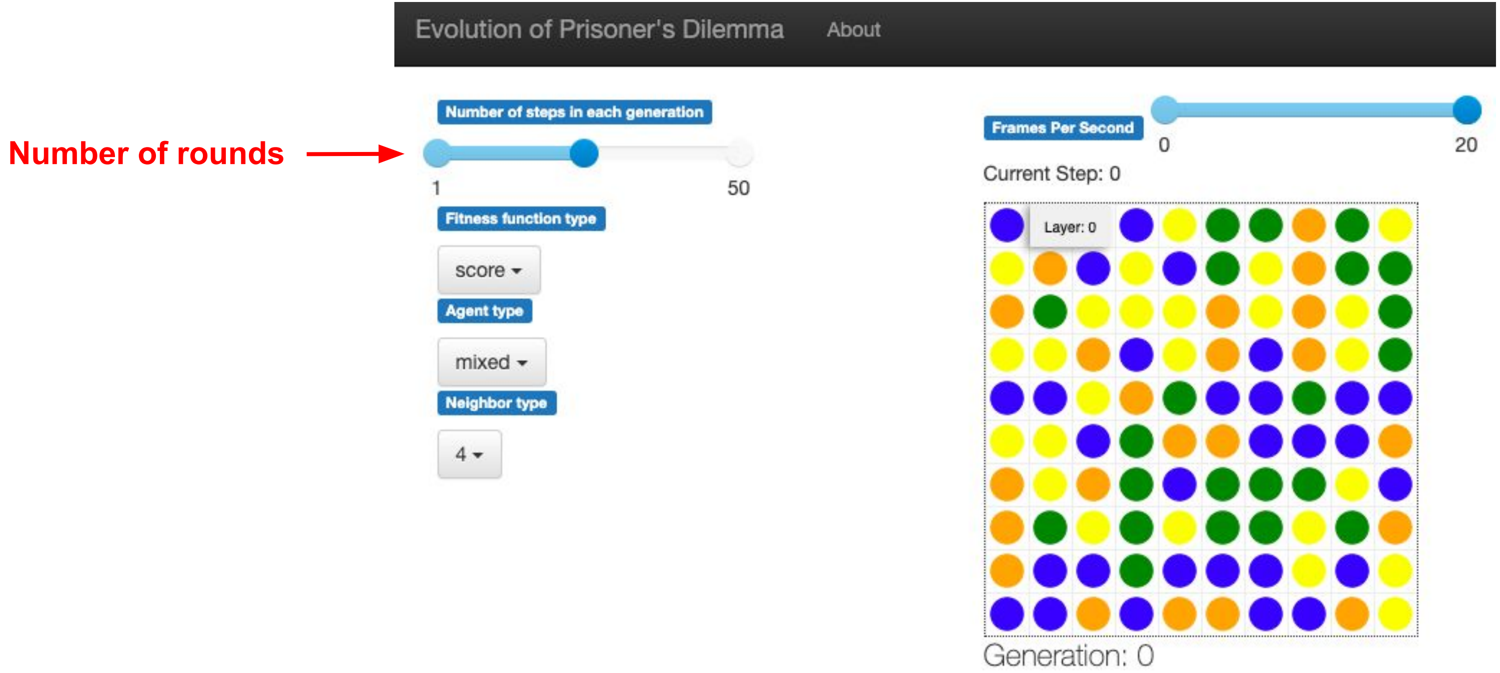


Config.py

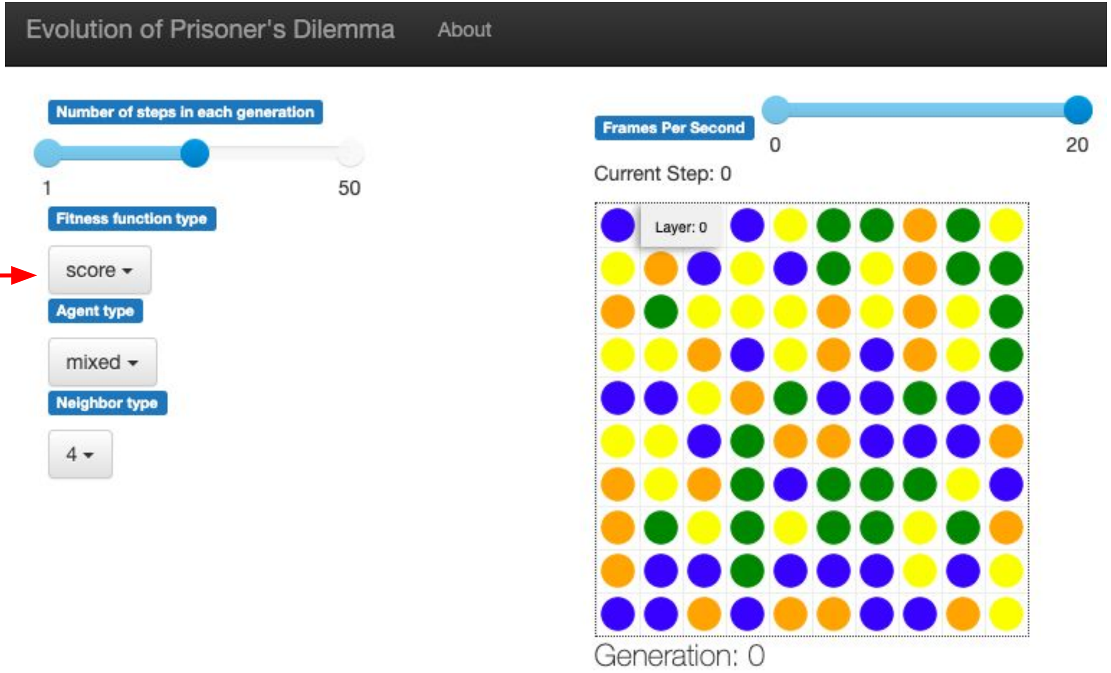
```python
# 0 cooperating, 1 defecting
# Map (my_action, other_action) to payoff
PAYOFF_MAP = [
    [3, 0],
    [5, 1],
]

# Neighbors of each agent: can be 8 (everyone around), 4 (left right top bottom), or 2 (left right)
NEIGHBOR_TYPE = 4
NEIGHBOR_RADIUS = 1  # interaction radius

NUM_SUBSTEPS = 10  # rounds in a generation

# Genetic Algorithm parameters
USE_LOCAL_GA = True  # weather to use the localized version of GA
FITNESS_MULTIPLIER = 2  # parameter used for scaling fitness function, see P15 of the java manual
EPS = 1e-3  # for fitness scaling
MUT_PROB = 0.2  # mutation probability for NeuralAgent and StringAgent
MUT_STRENGTH = 1.0  # how strong to perturb the weights of NeuralAgent
TFT_REPRODUCABLE = False  # does TFT participate in the reproduction
NEURAL_REPRODUCABLE = True  # does NeuralAgent participate in the reproduction
STRING_REPRODUCABLE = True  # does StringAgent participate in the reproduction

# Network topology for NeuralAgent
# e.g. [6, 1] means a network with 6 inputs (i.e. memory size 3), fully connected to the output
DEFAULT_NEURAL_STRUCTURE = [6, 1]

# StringAgent's memory size
MEM_LEN = 2
```

```python
# StringAgent's memory size
MEM_LEN = 2

# Settings for PDModel
DEFAULT_WIDTH = 10
DEFAULT_HEIGHT = 10
TORUS_GRID = True
CANVAS_DX = 30

# Random seeds
MESA_SEED = 4
NUMPY_SEED = 3

# The way agents are rendered in the We
# 'agent_type': the colors indicate whi
# 'defecting_ratio': visualize how defe
# 'inherited_attr': visualize who are t
VISUALIZE_GRID_TYPE = 'inherited_attr'
```

Website application via Mesa package



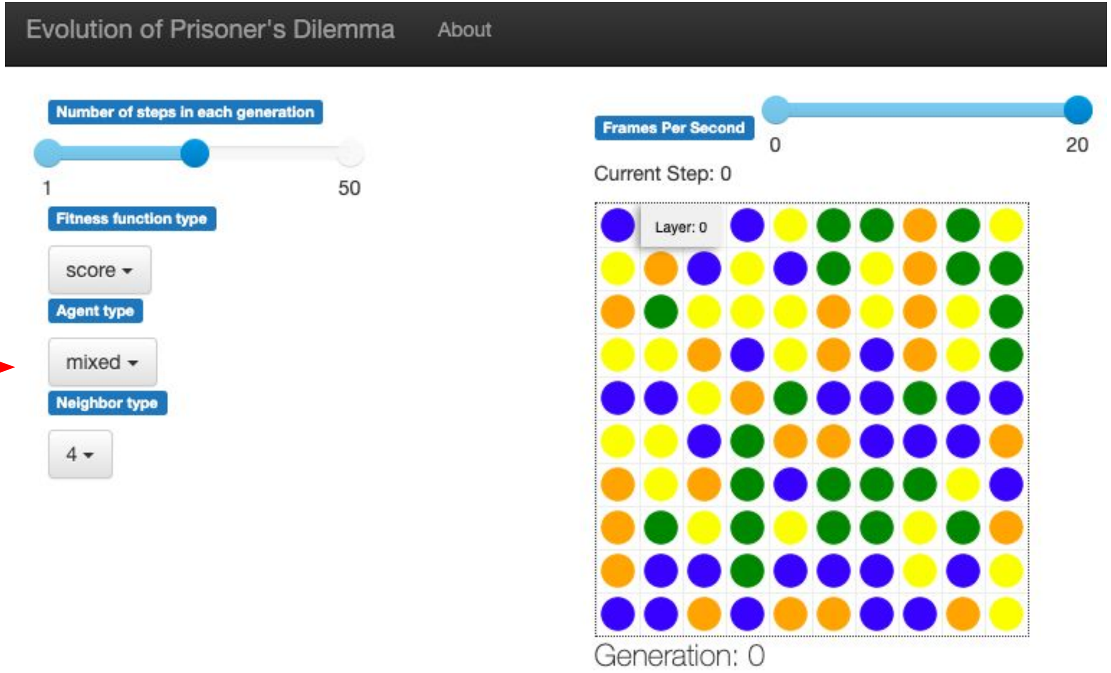
Website application via Mesa package



Fitness function type

score
cooperating_ratio
defecting_ratio

Website application via Mesa package



Agent type

neural

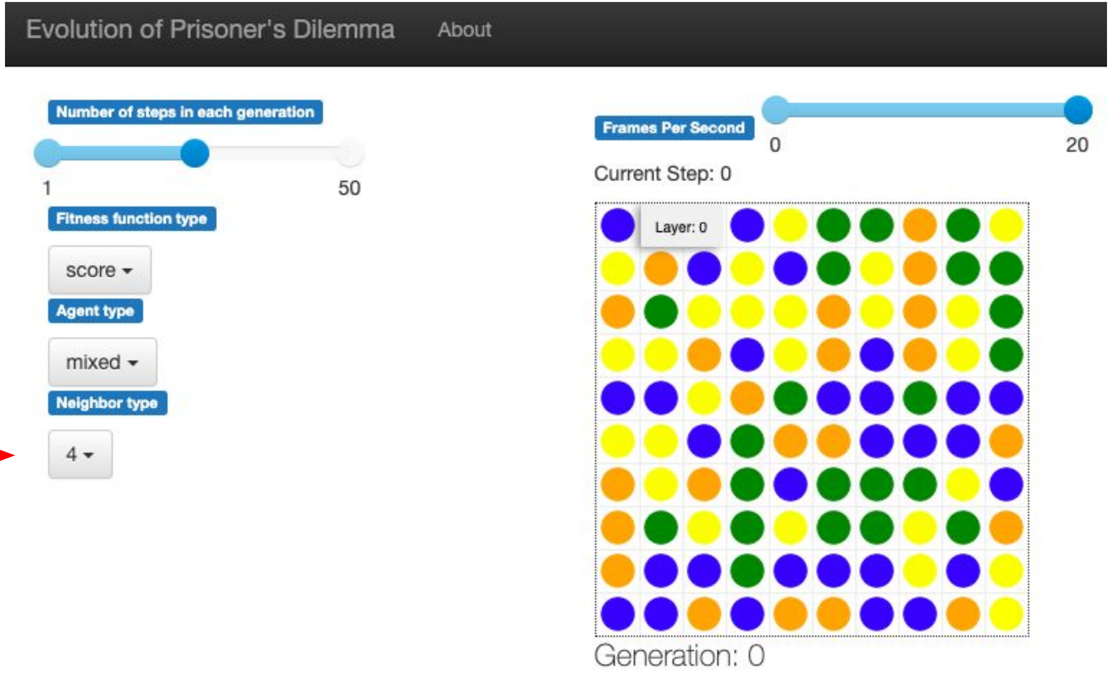
string

tit_for_tat

simple

mixed

Website application via Mesa package



Number of neighbors

8

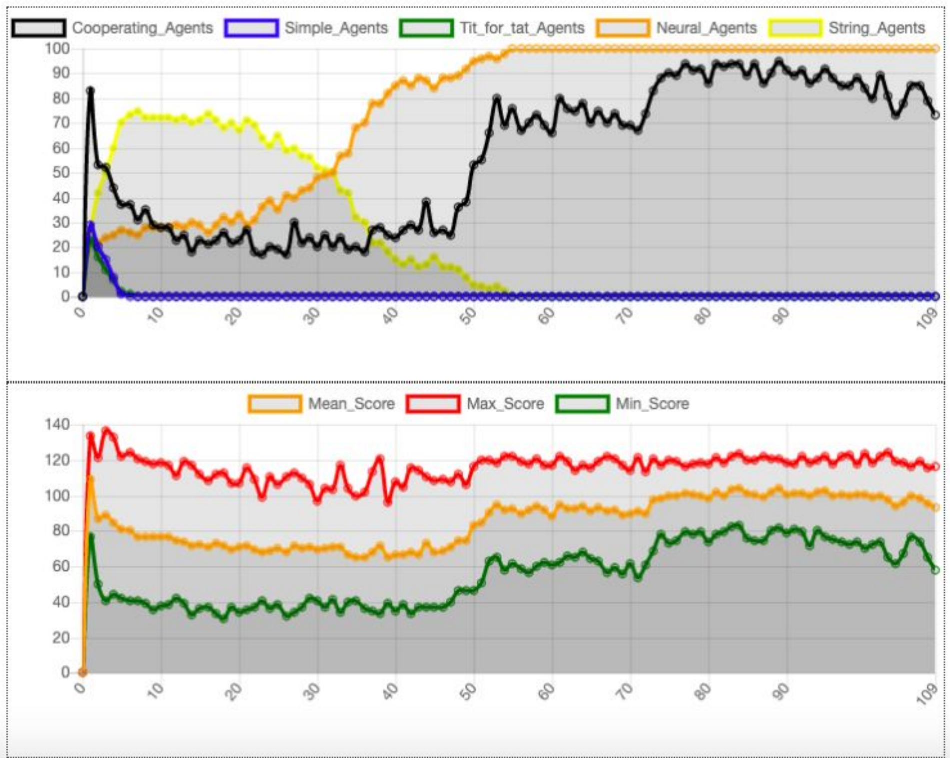
4

2

Website application via Mesa package

Generation: 109 Fittest feature vector: 0.4918, 0.6789

Distribution of agents



Distribution of scores

Execution panel

Results

Agents have to learn TFT for cooperation

Cooperation could be achieved by adding punishers

Local interaction + reproduction are important for cooperation

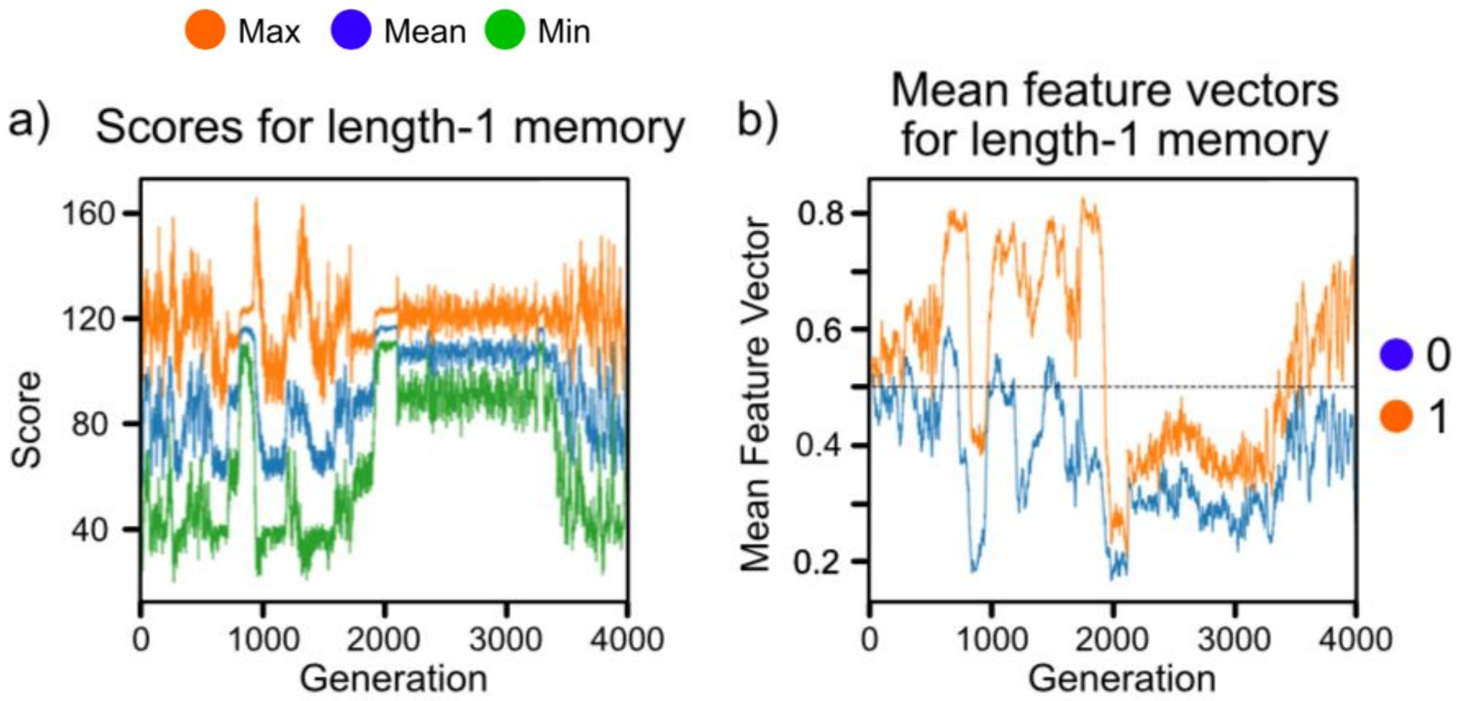
Larger Agent memory leads to stable cooperation

Questions

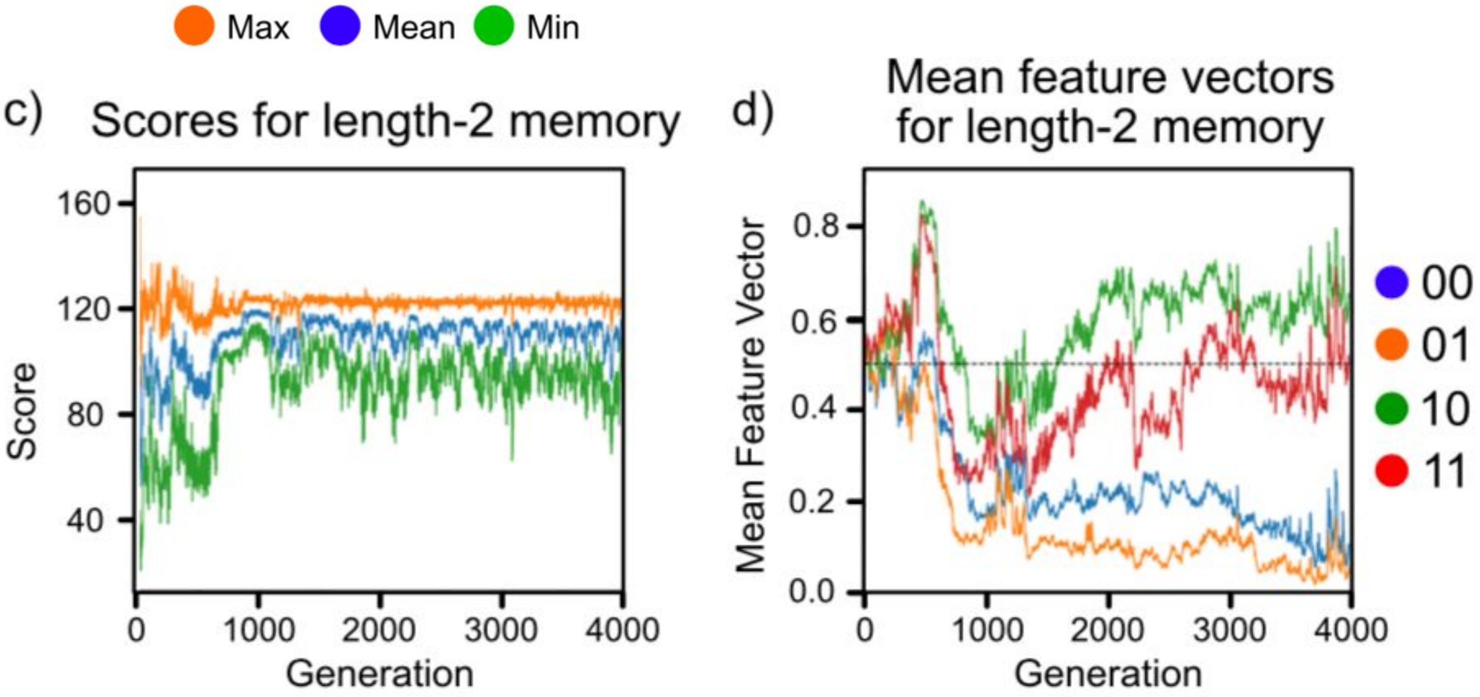
References

1. R. M. Axelrod and W. D. Hamilton, *The Evolution of Cooperation*. Basic Books, 1984, isbn: 0-465-02121-2.
2. R. Axelrod, *The Evolution of Strategies in the Iterated Prisoner’s Dilemma*. Lawrence Davis, London: Pitman, and Los Altos, CA: Morgan Kaufman, 1987, pages 32–41.
3. A. Errity, *Evolving Strategies for the Prisoner’s Dilemma*, 2003.
4. J. H. Holland, *Genetic Algorithms*, 1. Scientific American, a division of Nature America, Inc., 1992, volume 267, pages 66–73. url: http://www.jstor.org/stable/24939139.

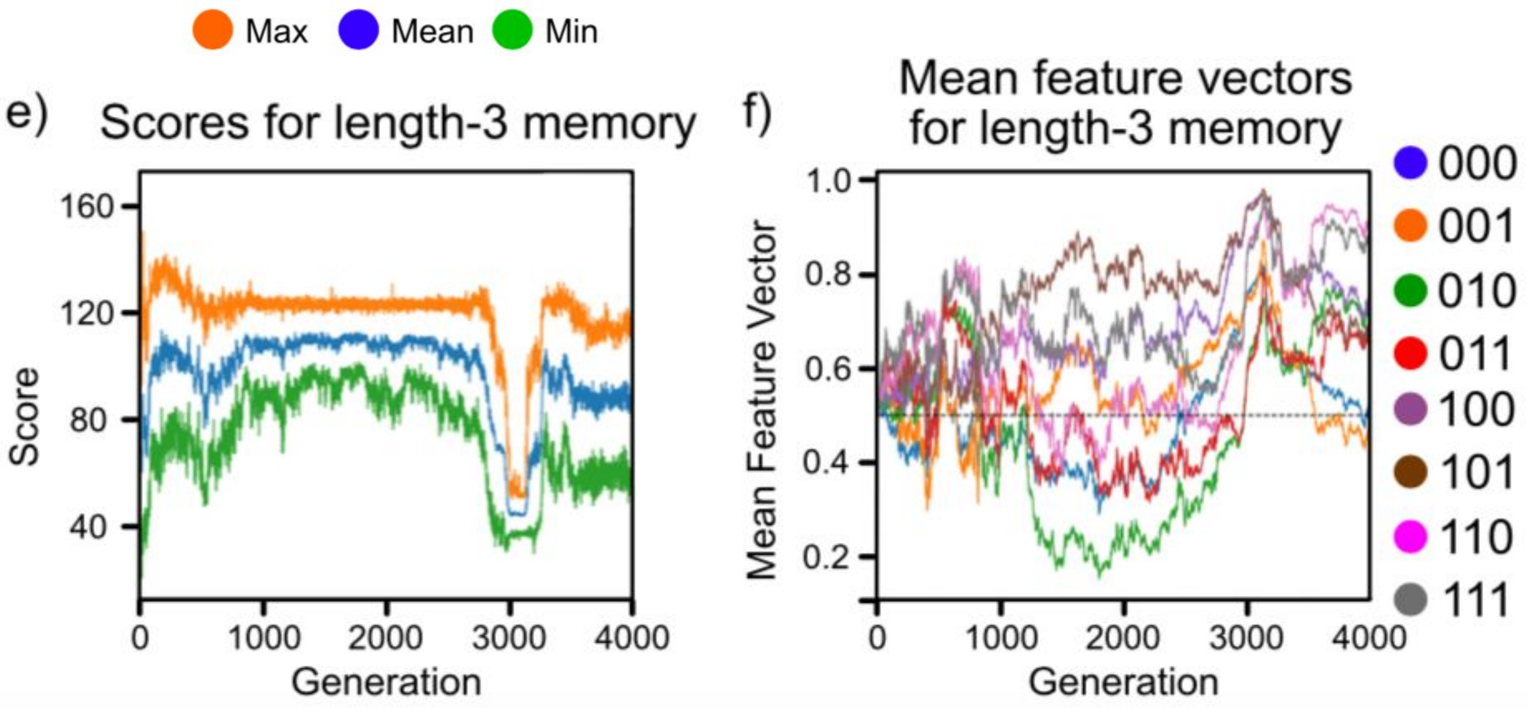
Memory-length tests



Memory-length tests

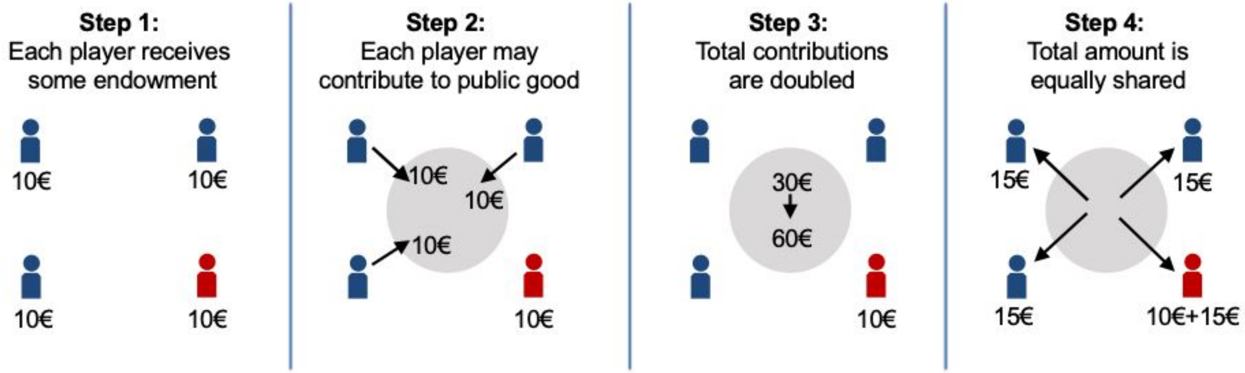


Memory-length tests



Connection to society?

- Public Good Game



- Examples: Tax evasion, climate change, cleaning shared apartment, Cross-Code-Checking
- Defectors lead to tragedy of the common
- Punishment and Reputation key for avoiding the tragedy of the commons