ETH zürich

The Emergence of Cooperation with Stationary Leadership

Jens Hunhevicz, Jonas Stolle, Sophie Kerstan & Pietro Griffa

Agent-Based Modeling and Social System Simulation FS19

Introduction: Relevance of Studying Cooperation

Defection is tempting because it guaranties «the largest payoff regardless of what the others are doing». (Dercole et al., 2019)

In the real-world, however, people accumulate in social groups where cooperation is common…

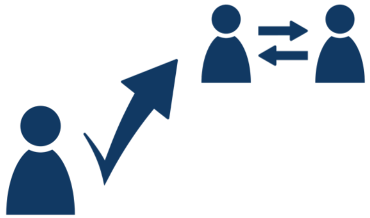
…and even companies cooperate to boost their competitive advantage and flock together in certain areas.

Theoretical Background

Game theory as the underlying framework:
Prisoners Dilemma Game (PDG), Public Goods Game (PGG), …

Mechanisms protecting and partly fostering cooperation implemented in models within the literatur

REPUTATION



IMITATION & IMIGRATION
(Helbing et al., 2011)

VOLUNTEERING

COSTLY PUNISHMENT
(Bahbouhi & Moussa, 2018)

Numerous others

Project Goals & Outcome Expectations

Overarching Goal

To implement **different mechanisms** and their **combinations** into an agent-based model in order to investigate **what promotes cooperation** within the Prisoner's Dilemma Game on a spatial grid

Sub-Goals

1. Replicate findings from Helbing et al. (2011):
   Unconditional imitation, success-driven migration, strategic noise
   → high levels of cooperation)

2. Extend model by adding "exemplary stationary leadership"
   = ALL C strategy, punishment of defection and reward of cooperation

3. Test if exemplary leadership has additional beneficial effects for cooperation

Expectations

Clusters formerly developing randomly will evolve around leaders

Leadership will show positive effects on the number of cooperators beyond the effect of imitation & immigration

(Parts of) the Model

Imitation

focal player – payoff comparison – strategy adoption in case of positive payoff difference

Success-driven migration

Mobility range M – test interactions – immitation

Leadership

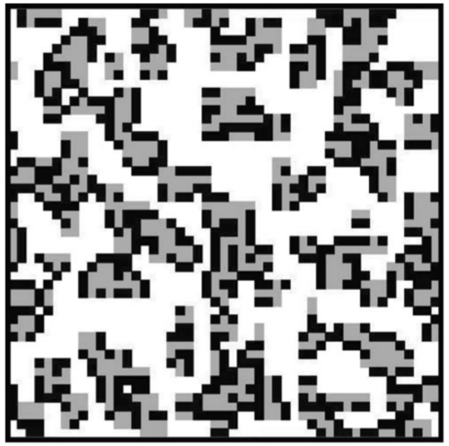
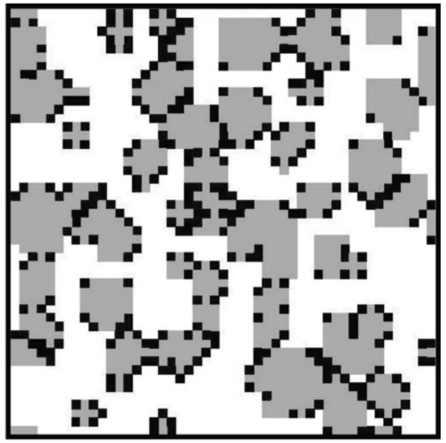
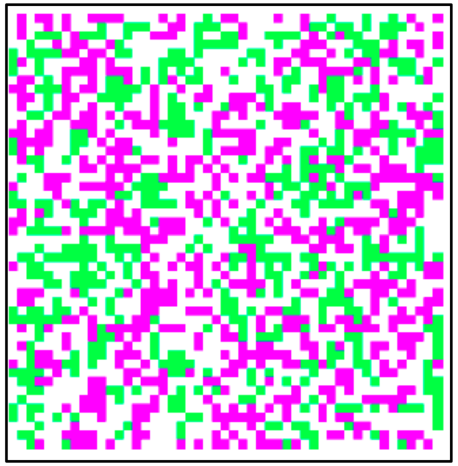
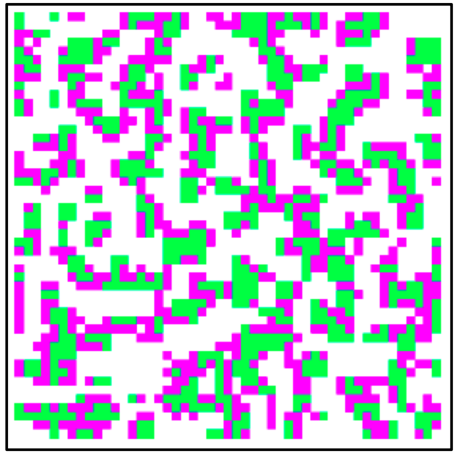
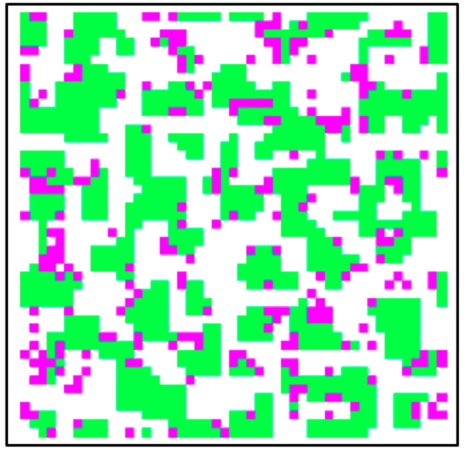
Random placement in grid – always cooperate

$$P = P_{ij} = \begin{pmatrix} P_{11} & P_{12} & \boldsymbol{P_{13}} \\ P_{21} & P_{22} & \boldsymbol{P_{23}} \\ \boldsymbol{P_{31}} & \boldsymbol{P_{32}} & \boldsymbol{P_{33}} \end{pmatrix} = \begin{pmatrix} R & S & \boldsymbol{x * R} \\ T & P & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{S} & \boldsymbol{R} \end{pmatrix}$$

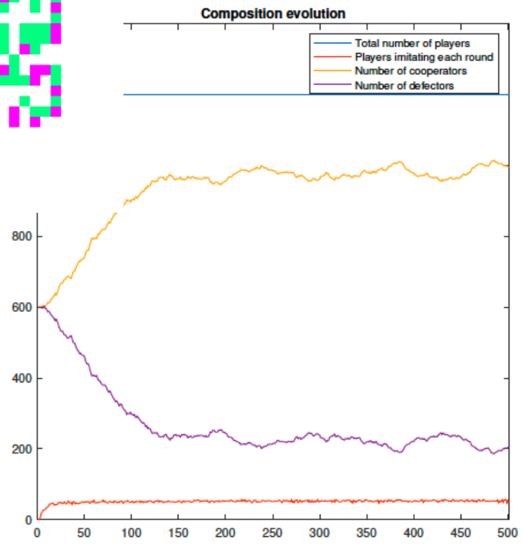
Noise

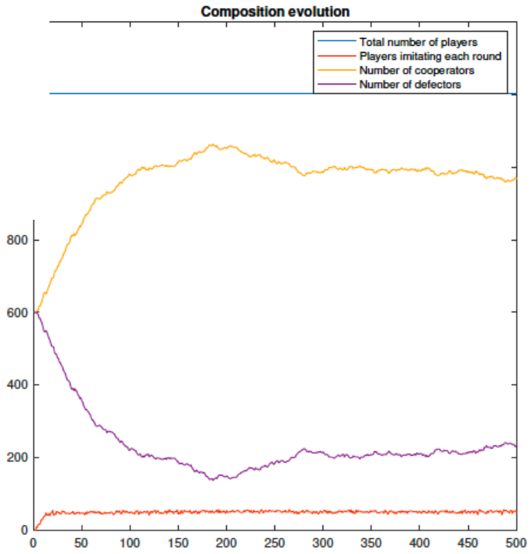
Random strategy mutations with proabability q

Results I: Replication

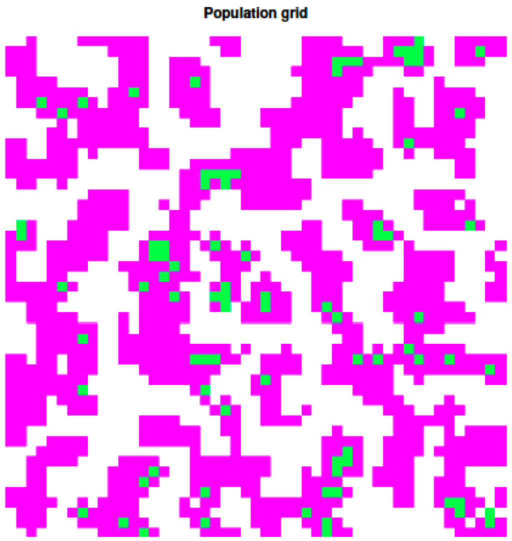
	Imitation only	Migration only	Both
Helbing et al. (2011)			
Our results			

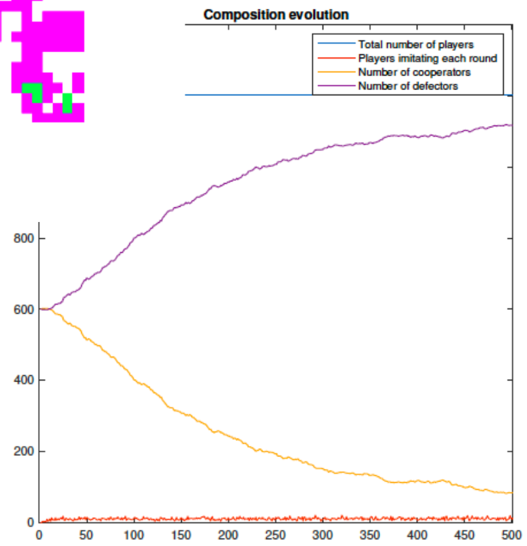
Results II:
High Levels of Migration/Immitation with and without Leadership

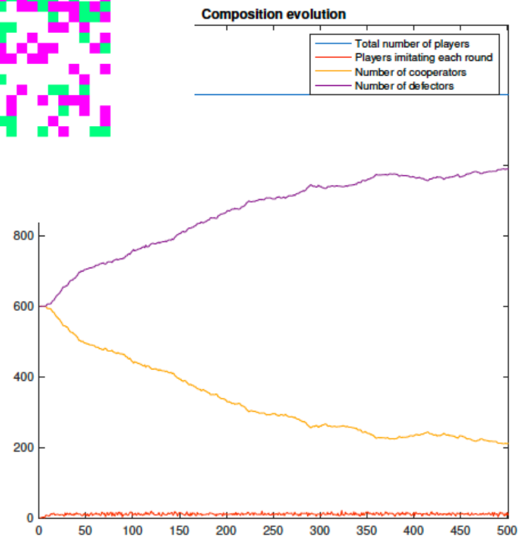




Results III:
Low Levels of Migration/Immitation versus Leadership







Discussion (Selection)

1. Cooperation evolves around leaders
2. Leadership keeps the dominating number of defectors at bay, protecting cooperation in scenarios where imitation and immigration fail as mechanisms

Limitations

- We adjusted the payoff matrix quite drastically

$$P = P_{ij} = \begin{pmatrix} P_{11} & P_{12} & \boldsymbol{P_{13}} \\ P_{21} & P_{22} & \boldsymbol{P_{23}} \\ \boldsymbol{P_{31}} & \boldsymbol{P_{32}} & \boldsymbol{P_{33}} \end{pmatrix} = \begin{pmatrix} R & S & \boldsymbol{x * R} \\ T & P & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{S} & \boldsymbol{R} \end{pmatrix}$$

- One could argue that the outcomes of leadership are rather related to reward an punishment, not to the “exemplary behaviour” of leaders

Results presented are likely only a fraction of possible interesting dynamics and outcomes that can be created with our model

Summary & Outlook

Summary

- Replicate and extend findings from Helbing et al. (2011) ✓
- Add “exemplary stationary leadership” ✓
- Test if exemplary leadership has additional beneficial effects ✓
- Clusters formerly developing randomly will evolve around leaders (✓)
- Leadership will show positive effects on the number of cooperators beyond the effect of imitation & immigration (✓)

Outlook

- Testing all combinations separately and systematically
- Testing different model parameter configurations
- Including other/even more mechanisms
- Implementation into other structures (networks) or applicationto other games (e.g. Hawk/Dove, Snowdrift, …)
- Conduction of lab experiments on the basis of the model
- …

Q&A

References

Dirk Helbing, Wenjian Yu, and Heiko Rauhut. Self-organization and emergence in social systems: Modeling the coevolution of social environments and cooperative behavior. Journal of Mathematical Sociology, 35(1-3):177–208, 2011.

Jalal Eddine Bahbouhi and Najem Moussa. A graph-based model for public goods with leaderships. Applied Mathematics and Computation, 349:53–61, 2019.

Backup Slide I: Pseudo-Code

```
input  : Grid of size L × L, density ρ, number of time steps t, sample size N,
         neighborhood dimension m, mobility range M, initial ratio of
         cooperators, payoffs T/R/P/S, migration true/false, imitation
         true/false, leadership true/false, noise true/false
output: The final spatial grid with cooperators and defectors after all interactions.
initialization of the grid calling function init() with L and ρ;
for number of time steps t do
    call function migration();
    for sample size N do
        if leadership then
            stationary leaders always cooperate;
        end
        1) play the Prisoners Dilemma game with T/R/P/S and update payoffs;
    end
    2) sum up payoffs in each neighborhood m calling function
     neighborhood_watch();
    if migration then
        3) do test-interactions on empty slots within mobility range M;
        if an empty slot has a higher payoff then
            4) move to empty slot with higher payoff in mobility range M;
        end
    end
    if imitation then
        5) imitate strategy of highest payoff within neighborhood m calling
         function imitate();
    end
    if noise then
        6) apply noise calling function noise();
    end
    update data and grid;
end
plot grid by calling function plot_pop();
```

Algorithm 1: *high-level pseudo-code abstraction*