Progress report:
Simulations with simple learning routine

Madis Ollikainen

June 15, 2015

1 Learning schema & strategy updating

I took the code for the single loop simulation and changed the strategy updating mechanism, such that it doesn't any more use Nash equilibrium, but rather a crude schema trying to imitate learning. Basically the agents now have a memory and after each turn they assign points to the strategy they used. These points are used in the next round for choosing a new strategy.

I introduced an array of doubles called *memory*, that holds a value for each possible strategies for each agent. Thus the array is of length $N_{agents} \times N_{strategies}$. If we wish to make simulations with a lot of agents using a fine grained semi-smooth strategy (contribution) interval, than this array might become to big. But at the moment it was OK for my laptop and had plenty of memory for more.

All entries in *memory* are initialised to 1. After each round t each agents i updates its memory entry M_ij for the strategy j that they used in that round. The update $M_{ij}(t) \to M_{ij}(t+1)$ is done as

$$M_{ij}(t+1) = M_{ij}(t)\left(1 + \xi \frac{w_i(t) - w_i(t-1)}{w_i(t-1)}\right),$$

where $w_i(t)$ is the wealth of agent i after round t and ξ is a pre-defined learning factor, which can be set in the configuration file under the *mUpDate* variable.

Before each round the strategies of all agents are updated using the entries in the array memory array M. The updating schema has the same probabilistic structure as it had in the previous simulations. The probabilities of each strategy j for agent i are calculated as

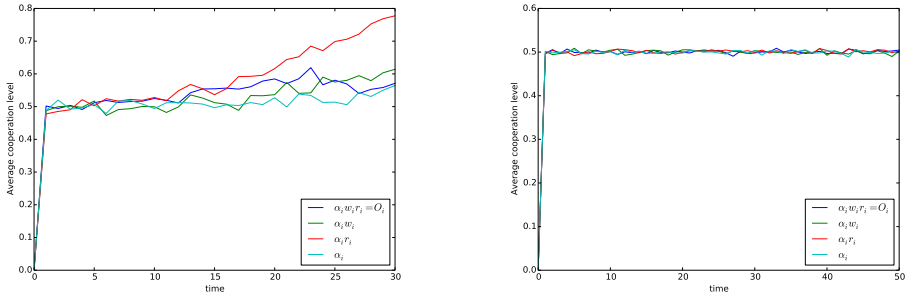
$$p_{ij} = \frac{e^{\beta M_{ij}}}{\sum_k e^{\beta M_{ik}}}.$$

2 Test simulations

2.1 Setup

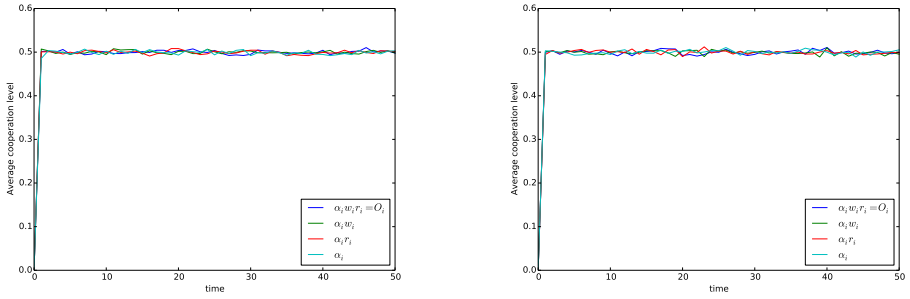
I ran three test simulation with $N_{agents} = 5000$ and $N_{strategies} = 101$ and $\xi = 1.0, 0.1, 0.01$. Each simulation lasted $t = 50$ time steps (using $\xi = 1.0$ the exponents exploded again after 70 or so time steps). For comparison I also ran a simulation with the Nash equilibrium strategy changing. But as that simulation schema take a lot longer to calculate, then I used just $N_{agents} = 500$ and also only for $t = 30$ time steps, as after that the exponent might explod. Other parameter were the same as in the *dummy.conf* (in which ever directory you look ...).

2.2 Results: plots



(a) Nash eq. comparison

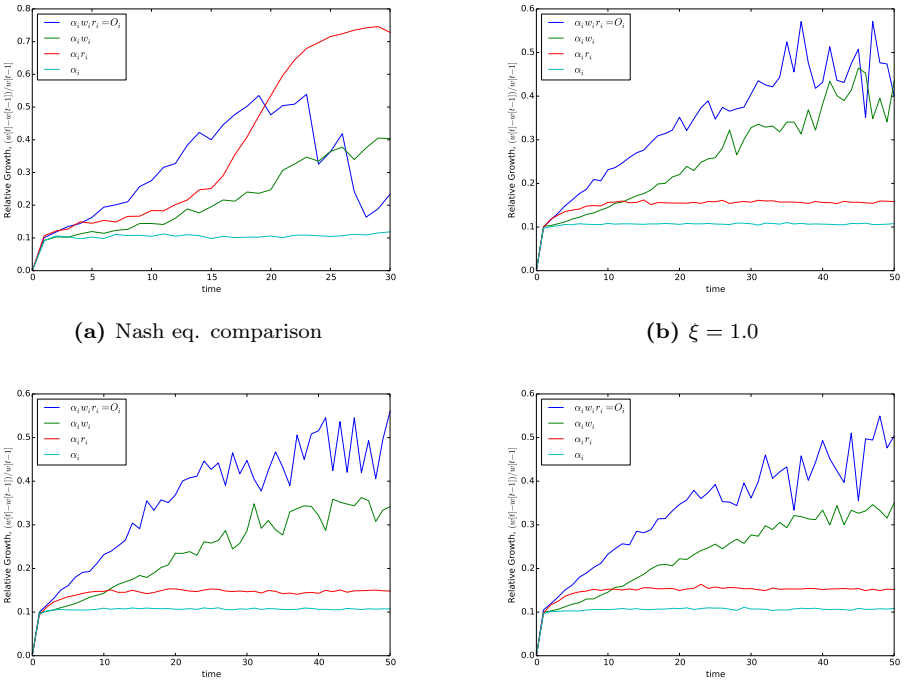
(b) $\xi = 1.0$



(c) $\xi = 0.1$

(d) $\xi = 0.01$

Figure 1: Average co-operation levels for different ranking systems for different simulation setups.



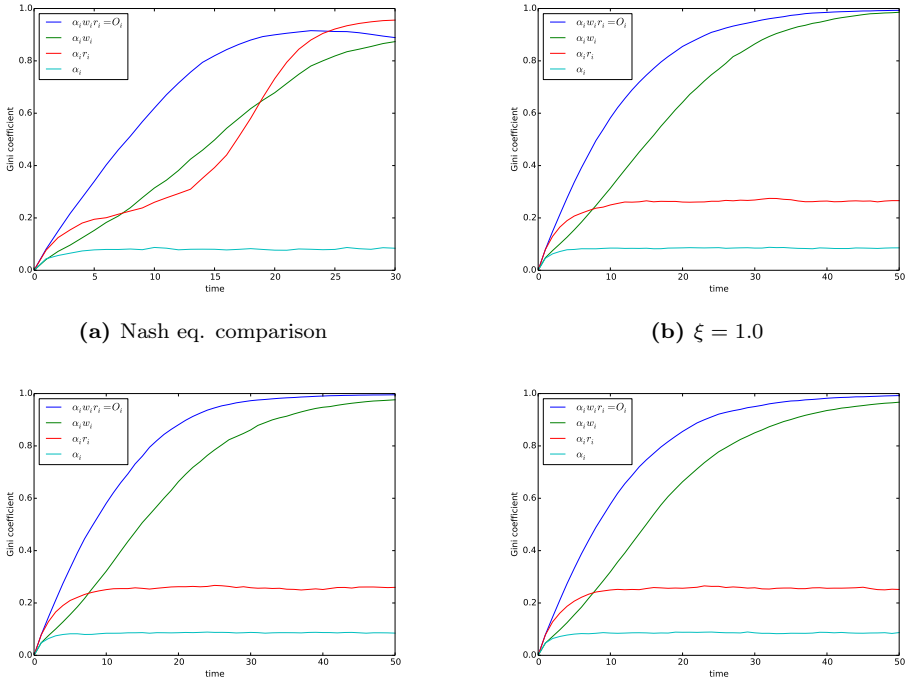
(a) Nash eq. comparison

(b) $\xi = 1.0$

(c) $\xi = 0.1$

(d) $\xi = 0.01$

Figure 2: Growth of wealth of the whole 'society' for different ranking systems for different simulation setups.



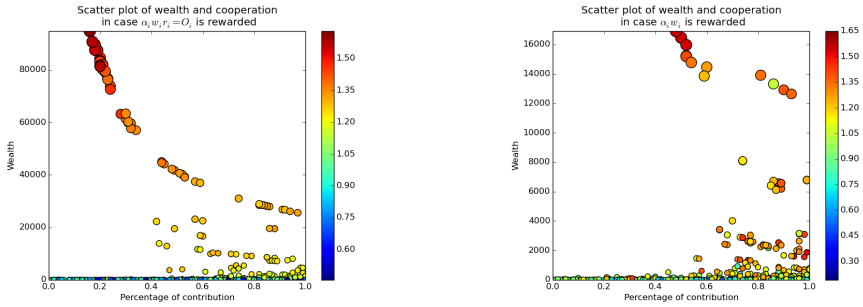
(a) Nash eq. comparison

(b) $\xi = 1.0$

(c) $\xi = 0.1$

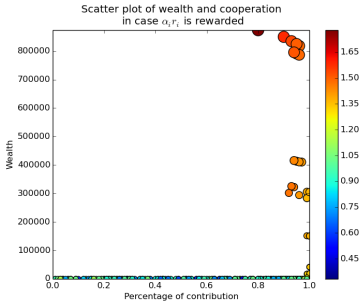
(d) $\xi = 0.01$

Figure 3: Gini coefficient for different ranking systems for different simulation setups.

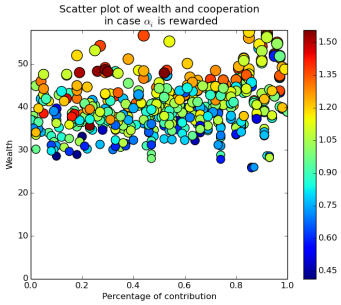


(a) Ranking 1

(b) Ranking 2

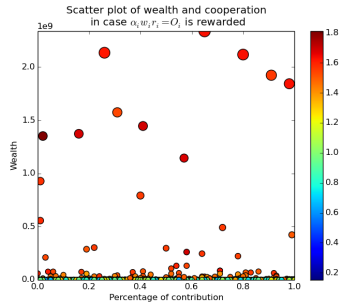


(c) Ranking 3

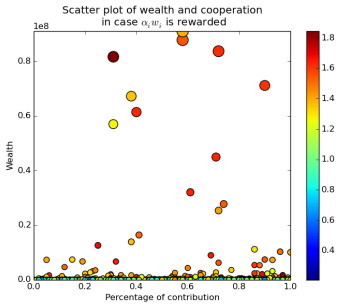


(d) Ranking 4

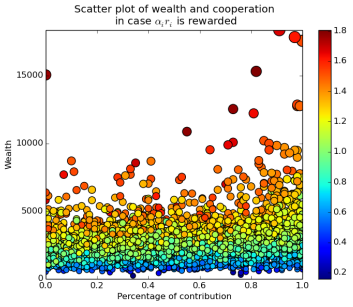
Figure 4: Scatter plots for the final time step for Nash eq. comparison.



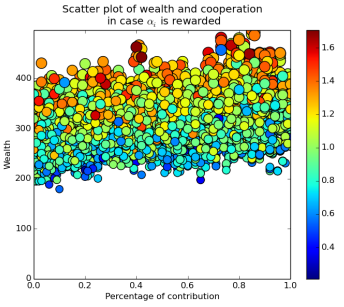
(a) Ranking 1



(b) Ranking 2

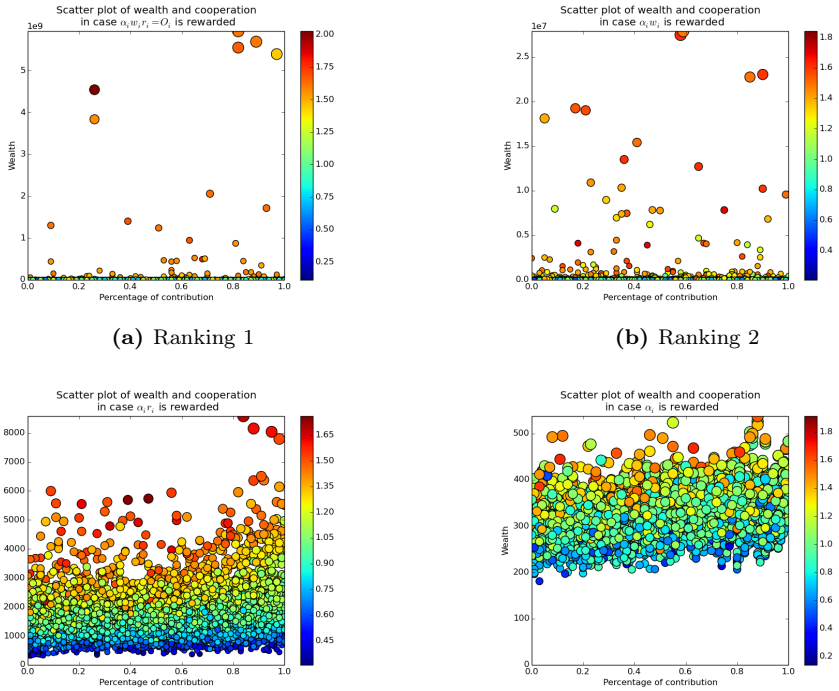


(c) Ranking 3



(d) Ranking 4

Figure 5: Scatter plots for the final time step for $\xi = 1.0$



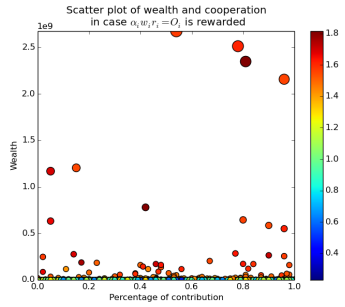
(a) Ranking 1

(b) Ranking 2

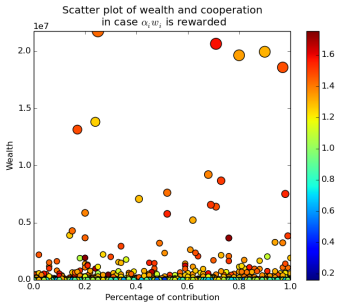
(c) Ranking 3

(d) Ranking 4

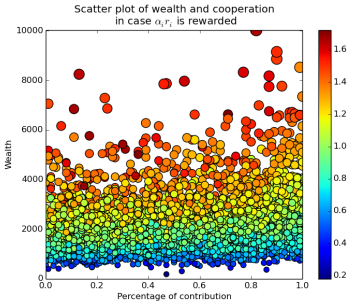
Figure 6: Scatter plots for the final time step for $\xi = 0.1$



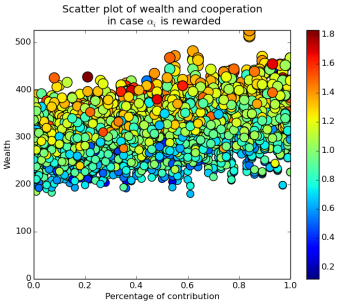
(a) Ranking 1



(b) Ranking 2



(c) Ranking 3



(d) Ranking 4

Figure 7: Scatter plots for the final time step for $\xi = 0.01$