Imitation and cooperation in different helping games: Accompanying material for

Giangiacomo Bravo (2008)
Imitation and Cooperation in Different Helping Games
Journal of Artificial Societies and Social Simulation vol. 11, no. 1 8
<http://jasss.soc.surrey.ac.uk/11/1/8.html>

Giangiacomo Bravo
Dipartimento di Studi Sociali
Università di Brescia

The model has been implemented using Netlogo 3.1 (http://ccl.northwestern.edu/netlogo/). After setting the simulation parameters and the turtle strategies, the main routine for all the no-move experimental condition is

```
to go
  help-others ;(i)
  imitate ;(ii / iii)
  update-globals ;(iv)
  do-plots ;(iv)
  if time = maxtime [stop]
end
```

while a "move-imitate" procedure (iii) replaces the "imitate" one in all the experimental conditions allowing agents to move.

(i) The "help-others" procedure implements the helping game (HG). For the private HG we have

```
to help-others
  ask turtles
    [let gain benefit * count (turtles-on neighbors) with [strategy = 1]
    set payoff payoff + gain - strategy * cost * count (turtles-on
     neighbors)
    ]
end
```

where strategy = 1 means cooperation, while strategy = 0 means defection. The variables "benefit" and "cost" represent *b* and *c* respectively. The payoff function for the public HG is simply

```
    set payoff payoff + gain - strategy * cost
```

(ii) The “imitate” procedure depends on the imitation condition. For the M condition we have:

```
to imitate
  ask turtles
   [let n-neighbors count (turtles-on neighbors)
   let tot-npayoff sum values-from (turtles-on neighbors) [payoff]
   let mean-payoff (tot-npayoff / n-neighbors)
   if payoff < mean-payoff
     [let similar-nearby count (turtles-on neighbors)
      with [strategy = strategy-of myself]
     let other-nearby count (turtles-on neighbors)
      with [strategy != strategy-of myself]
     if other-nearby > similar-nearby
       [ifelse strategy = 0
         [set strategy 1
         set color green
         ]
         [set strategy 0
         set color red
         ]
       ]
     ]
   ]
end
```

Notice that defectors are marked in red, while cooperators are marked in green (e.g. Figure 1 in the main text). For the S condition the procedure is

```
to imitate
  ask turtles
   [let n-neighbors count (turtles-on neighbors)
   let tot-npayoff sum values-from (turtles-on neighbors) [payoff]
   let mean-payoff (tot-npayoff / n-neighbors)
   if payoff < mean-payoff
     [let n-coop count (turtles-on neighbors) with [strategy = 1]
     let n-def count (turtles-on neighbors) with [strategy = 0]
     if n-coop = 0
       [set strategy-of self 0
       set color red
       ]
       if n-def = 0
         [set strategy-of self 1
         set color green
         ]
       if n-coop > 0 and n-def > 0
        [let coop-payoff sum values-from (turtles-on neighbors)
         with [strategy = 1]  [payoff]
        let mean-coop-payoff coop-payoff / n-coop
        let def-payoff sum values-from (turtles-on neighbors)
         with [strategy = 0]  [payoff]
        let mean-def-payoff def-payoff / n-def
        if  mean-def-payoff > mean-coop-payoff
          [set strategy-of self 0
          set color red
          ]
        if  mean-def-payoff < mean-coop-payoff
          [set strategy-of self 1
          set color green
          ]
        ]
```

```
      ]
    ]
end
```

Finally, for the MS condition the “imitate” procedure is simply

```
to imitate
  ask turtles
   [let max-payoff max values-from (turtles-on neighbors) [payoff]
   if max-payoff > payoff
     [let new-strategy value-from one-of (turtles-on neighbors)
     with [payoff = max-payoff] [strategy]
     set strategy new-strategy
     if strategy = 1 [set color green]
     if strategy = 0 [set color red]
     ]
   ]
end
```

(iii) In the move condition, the “move-imitate” procedure selects whether “unsatisfied” agents chose to move or to imitate:

```
to move-imitate
   ask turtles
   [let n-neighbors count (turtles-on neighbors)
   ifelse n-neighbors > 0
     [let tot-npayoff sum values-from (turtles-on neighbors) [payoff]
     let mean-payoff (tot-npayoff / n-neighbors)
     if payoff < mean-payoff
       [ifelse random 1001 < m
         [move]
         [imitate]
       ]
     ]
     [move]
   ]
end
```

where $m \in \{1, \ldots , 999\}$ is the propensity to move of each agent. The “imitate” procedure depends, as below, from the imitation condition, while the “move” procedure is

```
to move
  ask turtles
   [let newx xcor + ((random 2) * 2 - 1)
   if newx < 0 [set newx 31]
   if newx > 31 [set newx 0]
   let newy ycor + ((random 2) * 2 - 1)
   if newy < 0 [set newy 31]
   if newy > 31 [set newy 0]
   let test any? turtles-on (patch newx newy)
   if test = false
     [set xcor newx
     set ycor newy
     ]
   ]
end
```

(iv) The “update-globals” procedure updates the global variables after each round and the “do-plots” one plots the selected indicators.