Full Model Description
Op_Cost_Walk_Away.nlogo
VERSION 1.0 (November, 2018)

Model Description

This model description follows the guidelines of the ODD protocol (Grimm et al. 2010). We note at the outset that the model used to collect the data presented in our publication runs on NetLogo 6.0.2. The code may need to be modified to run on later versions of NetLogo.

Purpose

Aktipis (2004) implemented a spatially explicit, iterated prisoner's dilemma simulation in Starlogo 1.2.2 to test whether unconditional cooperators (always play 'C') could persist among, or even outcompete, defectors if cooperators were able to simply "walk away" from partners who defected upon them (played 'D'). This model is a replication and extension of Aktipis' basic simulation experiment, which included just four strategies for playing the prisoner's dilemma (described below). In general, the model allows one to explore the evolution of cooperation in the spatial iterated prisoner's dilemma under a range of socio-ecological conditions. More specifically, the purpose of the model is to address the following research question: how does population density, error rate, and offspring dispersal affect the success of "walk away" cooperators in the spatial iterated prisoner's dilemma?

Entities, state variables, and scales

The model has two types of agent entities: mobile players (turtles in NetLogo) and square cells (patches in NetLogo) on a toroidal lattice. Cell size is not ecologically meaningful. There are four types of players, each represented by a different breed. Each type corresponds to one of four strategies (see below). Players are characterized by many state variables (described below). The number of players that can occupy a given patch is unrestricted. The dimensions of the lattice and the number of players of each type can be adjusted. A time step represents the time required to either play one round of the prisoner's dilemma and move one cell to one's left or right (relative to one's heading) in response to a defection or to simply move one cell to one's left or right (relative to one's heading) in search of partner.

Global Variables

seed
  * Specifies the value of random-seed. Each unique simulation run gets its own unique seed value. The seed values used in Premo and Brown (2019) are provided in the experiments available in BehaviorSpace.

world-max-xy
  * Sets the square dimensions of the toroidal lattice.
  * Modifying world-max-xy while holding constant the total number of players allows one to adjust population density without affecting the strength of genetic drift relative to the strength of selection.

* We used values of 11, 24, 49, and 99, which correspond to lattices of 12x12, 25x25, 50x50, and 100x100, respectively.

n-WACs, n-NCs, n-WADs, n-NDs
* Number of players of each breed present at the start of the simulation.

carrying-capacity (K)
* The maximum number of players allowed at the end of any time step during the simulation.
* Maintained by the enforce-carrying-capacity procedure (described below).
* We set carrying-capacity to 100 for all of our experiments.

error-prob (e)
* The probability that a cooperator (WAC, NC) will erroneously defect or a defector (WAD, ND) will erroneously cooperate when playing the prisoner's dilemma. This type of error has been described as the "trembling hand."
* We used values of 0.001, 0.01, and 0.1.
* Note that an error-prob of 0.5 means that there is no difference in strategy between a cooperator and a defector. Values greater than 0.5 mean that cooperators defect more often than they cooperate and defectors cooperate more often than they defect.

mu (μ)
* The probability that one's offspring inherits a breed other than the parent's breed due to mutation.
* We used values of 0, 0.001, and 0.01.

non-player-pay
* The amount added to the energy budget of each player that does not have a partner during the current time step.
* We held this constant at 0.

offspring-dispersal
* This variable defines how far afield from the parent an offspring disperses at birth.
* The model allows for one of two types of offspring dispersal:
  1. global: the offspring can be placed anywhere within the lattice
  2. local: the offspring can be placed anywhere within a local neighborhood of cells surrounding the parent. The parent's local neighborhood is defined as the cells within a radius of dispersal-radius, excluding the parent's cell.

dispersal-radius
* The extent of the boundary (in patch-widths) within which an offspring must be placed when offspring-dispersal = local.
* Premo and Brown (2019) holds dispersal-radius constant at 3.

plot-data
* This switch turns on visual plots of assortment, CV payoff/tick, % paired, mean payoff/tick, and mean % time playing game. The simulation runs faster when this switch is turned off.

Prisoner's dilemma payoffs
To be consistent with the assumptions of prisoner's dilemma the four PD payoffs must meet both of the following conditions: $T > R > P > S$, and $R > (T+S)/2$.

Temptation to cheat payoff (T)
* Payoff to ego when ego defects and partner cooperates.
* We used a value of 5, as in Aktipis (2004).

Reward for mutual cooperation payoff (R)
* Payoff to ego when ego cooperates and partner cooperates.
* We used a value of 3, as in Aktipis (2004).

Punishment for mutual defection (P)
* Payoff to ego when ego defects and partner defects.
* We used a value of 0, as in Aktipis (2004).

Sucker's payoff (S)
* Payoff to ego when ego cooperates and partner defects.
* We used a value of -1, as in Aktipis (2004).

Other global variables used for code verification, data analysis, or visualization (described further with comments in the source code):
total-exploitation, total-births, total-deaths, error-count, mutation-count, games-played-count, WAC-won, WAD-won, NC-won, ND-won, WAC-percent-paired, WAD-percent-paired, NC-percent-paired, ND-percent-paired, total-percent-paired, WAC-assortment, WAD-assortment, NC-assortment, ND-assortment, total-assortment, mean-cluster-size, sd-cluster-size

4 player types (i.e., four different "breeds" in NetLogo)
"Naive" Cooperator (breed=NCs)
* When playing prisoner's dilemma, cooperate with probability (1 - error-prob) and defect with probability error-prob. Do not move in response to a partner's defection.

"Naive" Defector (breed=NDs)
* When playing prisoner's dilemma, defect with probability (1 - error-prob) and cooperate with probability error-prob. Do not move in response to a partner's defection.

"Walk Away" Cooperator (breed=WACs)
* When playing prisoner's dilemma, cooperate with probability (1 - error-prob) and defect with probability error-prob. Respond to a partner's defection by moving one cell to one's left or right.

"Walk Away" Defector (breed=WADs)
* When playing prisoner's dilemma, defect with probability (1 - error-prob) and cooperate with probability error-prob. Respond to a partner's defection by moving one cell to one's left or right.

Player State Variables

xcor, ycor

* The x, y coordinate of each player marks its location on the lattice. The relevant scale for location in this model is the cell (or patch)—a player can only interact with other players that inhabit the same cell, and all player movement is between cells rather than within a cell.

heading

* Players are initialized with one of the four cardinal headings (0, 90, 180, 270).
* Heading is updated each time a player moves (the agent's heading is reset to face the direction in which it moved: 0, 90, 180, or 270).

energy

* This represents the energy budget of the player.
* At initialization, this value is pulled from uniform distribution bound by 0 and 50, not including 50.
* This dynamic state variable is updated under three conditions:
  1. As a result of playing the prisoner's dilemma, the relevant payoff is added to energy (note that the payoff can be a negative value)
  2. 10 units are subtracted from the energy budget of any player selected by the enforce-carrying-capacity submodel (described in more detail below)
  3. When a player reproduces, its current energy is split evenly with its new offspring

strategy

* 1 = cooperate (play "C") in the prisoner's dilemma
* 0 = defect (play "D") in the prisoner's dilemma
* The value set at the start of each time step corresponds to the player's breed (for WAC & NC, strategy = 1, and for WAD & ND, strategy = 0).
* The value set at the start of the time step changes only if a player makes an error. In the event of an error, cooperators change their strategy to 0 and defectors change their strategy to 1 (see more in **Go** description).

color

* Color is used to indicate the breed of the player: WACs = cyan, WADs = red, NCs = lime, and NDs = orange.
* Player color changes to grey or white when agents are paired during the course of a time step, but reverts to the breed's color at the beginning of each time step.
* Color is used only for visual debugging.

tried-to-pair

* The value of this variable indicates whether the player has already attempted to find a partner during the current time step. This variable serves as a fail-safe to ensure that each player cannot run the pair submodel more than once per time step.
* The value is set to 0 for all players at the start of each time step.
* When a player runs the pair submodel, its tried-to-pair is set to 1 regardless of whether the player finds a partner in its cell.

size
  * This static state variable is purely cosmetic. It is set to 2 for all players, regardless of breed.

partner
  * If an unpaired player (ego) finds another unpaired player in its cell, ego sets its partner to that player and that player sets its partner to ego. The agent set as ego's partner will serve as ego's opponent (and vice versa) when playing the prisoner's dilemma during the current time step.
  * All players set partner to "nobody" at the start of each time step. Thus, players do not remember their partners from the previous time step.

played-already
  * This variable prevents paired players from playing the prisoner's dilemma more than once per time step.
  * All players set played-already to 0 at beginning of each time step. Players set this value to 1 as soon as they play the prisoner's dilemma to ensure that they cannot play the PD more than once per time step.

moving
  * The value of this variable indicates whether the player should run the move submodel (described below) during the current time step.
  * 0 = do not move, 1 = move
  * All players set moving to 0 at the beginning of each time step.
  * A player sets moving to 1 under two conditions:
    1. The player did not have a partner during the current time step.
    2. The player is a WAC or WAD whose partner defected during play of the prisoner's dilemma.

potential-breed-list
  * The list of breeds that can be assigned to a parent's offspring as a result of mutation during reproduction.
  * The list contains the breeds present at the beginning of the simulation, minus the parent's breed. The maximum length of this list is 3, but it can be a short as 1.
  * The offspring is given one of the breeds on this list (chosen randomly) instead of its parent's breed.

Other player state variables used for data analysis and visualization:
Each player keeps track of the number of other players located within dispersal-radius (cluster-size). Players keep track of the cumulative number of times they play and the cumulative number of times they fail to find a partner. They also keep a running total of payoffs from play in the prisoner's dilemma. These data are used to calculate parameters used in three figures displayed on the interface: mean payoff/tick, CV payoff/tick, mean % time playing game.

Process overview and scheduling
The *go* procedure provides the schedule of procedures called during each time step. No player moves on to the next procedure until all eligible players complete the current procedure. The

"ask" primitive ensures that player order is randomized during each procedure. The names of submodels are also italicized. Each submodel is described in more detail below in the "Submodels" section of the model description.

At the start of the *go* procedure, all players are asked to reset the values of player state variables that may have been modified during the previous time step (e.g., color, partner, moving, tried-to-pair, and played-already) to their default values. Next, every player who does not already have a partner (partner = nobody) runs the *pair* submodel once. Every player that does not find a partner is asked to add 1 to its times-without-partner tally and add non-player-pay to its energy budget. Every player is then asked to compare a real number drawn randomly from a uniform distribution bound by 0 and 1 to error-prob. This constitutes the *error* submodel. If the random value is less than error-prob, the player is asked to change its strategy from 1 to 0 or vice versa (from 0 to 1 if its strategy is 0 at the start of the time step). Next, those players who have a partner (partner != nobody) and have not yet played the prisoner's dilemma in the current time step (played-already = 0) are asked to run the *play* submodel. After all eligible players finish the *play* submodel, those players whose energy has dropped to (or below) 0 die. Next, those players who need to move (moving = 1) are asked run the *move* submodel. Then, those players with energy greater than or equal to 100 are asked to *reproduce*. If the population size exceeds the carrying-capacity after all eligible players have reproduced, the observer runs the *enforce-carrying-capacity* submodel until the number of players equals carrying-capacity. Next, all cooperators (WACs, NCs) are asked to set strategy equal to 1 and all defectors (WADs, NDs) are asked to set strategy equal to 0. This ensures that errors made in the current time step are not carried forward into the next time step. The last few procedures in *go* concern data collection, plotting values to the interface, and enforcing the stop conditions of the simulation. When mu = 0, simulations stop when there is only one breed remaining in the population. When mu > 0, simulations stop after 500,000 time steps (ticks). The *go* procedure is repeated until the appropriate stop condition is met.

Design concepts

The *basic principle* addressed in this model is the robustness of contingent movement as a strategy that could promote the evolution of cooperation in a spatially explicit iterated prisoner's dilemma. Premo and Brown (2019) explores how various socio-ecological conditions affect the evolutionary success of "walking away" from defection in the spatially explicit prisoner's dilemma. We varied population density (by adjusting the size of the lattice while holding carrying capacity constant), the probability of committing an error while playing the prisoner's dilemma, the probability of mutation during reproduction, and the geographic extent of offspring dispersal.

As one might suspect, player *interaction* is a key component of this model. Players must be located on the same patch in order to play against each other in the prisoner's dilemma. Given that players employ a random walk (i.e., "sidestep") to search for PD partners, the likelihood of finding a partner to play is affected by population density.

As long as carrying capacity is sufficiently high to render drift a weak force of evolutionary change relative to the strength of selection, breeds that increase in frequency are evolutionarily

adaptive for a given socio-ecological context, which includes the relative frequency of other breeds. In short, the relative frequency of each breed is related to its relative *fitness*.

All players share the *objective* of finding a partner and playing the prisoner's dilemma in order to gain energy required to reproduce. One might say that WACs and WADs have an additional objective—to respond to partners that defect during the prisoner's dilemma. However, the ultimate *objective* for all players is to gain energy so as to reproduce as frequently as possible.

Players have limited *sensing* abilities. A player can sense other players that inhabit the same patch it inhabits.

The model is initiated with a user-defined seed, which affects the *stochastic* processes in the model. The model includes a number of stochastic processes, including the initial assortment of players on the lattice, their initial energy values, "trembling hand" errors in strategy, the direction of player movement, and mutation during reproduction. Also note that players are chosen randomly (and with replacement) by the enforce-carrying-capacity submodel.

Important *observations* in this model include the proportion of times each breed "wins" (i.e., evolves to fixation in the population at the expense of all other strategies included at the start of the simulation) when mu = 0, or is "successful" (represented by at least 95 agents after 500,000 time steps) in simulations with mu > 0. The model records many other types of observations that were not discussed in Premo and Brown (2019), some of which are displayed on the interface as monitors and graphs. Some important metrics include the relative frequencies of each breed, the total number of births and deaths, mean payoff per tick, and mean % of time spent playing the PD. Data collected intermittently during the simulation run is written to a file named "walk_away_data.txt" in the same folder that contains this nlogo file. NetLogo will prompt you for the directory in which to save the results from BehaviorSpace experiments. Simulations without mutation (mu = 0) run until just one breed remains. Simulations with mutation (mu > 0) run for 500,000 time steps.

Initialization

Table 1 provides the parameter values used to initialize the simulations reported in Premo and Brown (2019). These values can be found in the published paper as well as in the experiments saved in BehaviorSpace. At the start of each simulation, players are randomly distributed about the lattice, each player's energy is set to a unique random value drawn from a uniform distribution bound by 0 and 50 (not including 50), and each player's heading is set to one of the four cardinal points on the compass (0, 90, 180, and 270) with equal probability. Cooperators (NC and WAC) start with strategy=1 and defectors (ND and WAD) start with strategy=0.

Table 1. Parameter values used to initialize the simulations reported in Premo and Brown (2019).

Parameter	**Value(s)**
carrying-capacity, K	100
reward, R	3
temptation, T	5
sucker's payoff, S	-1
punishment, P	0
dispersal-radius	3
non-player-pay	0
world-max-xy	11, 24, 49, 99
error-prob, *e*	.001, .01, .1
mu, μ	0, .001, .01
offspring-dispersal	local, global

Replication

To replicate the findings presented in Premo and Brown (2019), first save the NetLogo model to your computer. Then run the experiments listed in BehaviorSpace, accessed from the Tools drop-down menu.

Input Data

The model does not require any input data or additional files to initialize a simulation run.

Submodels (listed in procedural order)

pair

At the start of this submodel, the player sets tried-to-pair equal to 1, which disallows them from running *pair* more than once per time step. The prisoner's dilemma is played between 2 partners. Partners are players located on the same cell in the lattice. Each player who does not have a partner and who hasn't already attempted to pair with a partner during the current time step, sets its partner to one of the other partnerless players on the same patch. The player identified as partner is then asked to set its partner to ego. These two players are now paired, and they will play the prisoner's dilemma against each other later in the time step. If there are no other players on ego's patch or if all of the other players on ego's patch already have partners, then ego will remain un-paired during the current time step. All players that do not find a partner to pair with set moving to 1, making them eligible to run *move* later in the time step.

error

Every agent (including those that are unpaired, though it doesn't matter in the end for them because they won't be playing the prisoner's dilemma during the current time step) is asked to compare a real number drawn randomly from a uniform distribution bound by 0 and 1 to error-prob. If the random value is less than error-prob, the player changes its strategy from 1 to 0 or from 0 to 1, as the case may be. Note that in either case the strategy is reset to correctly match

the agent's breed near the end of the time step, well after all paired agents have played the prisoner's dilemma.

play
Each player that has a partner (partner != nobody) and has not already played the prisoner's dilemma during the current time step (played-already=0) compares its strategy to the strategy of its partner and updates its energy by adding the relevant prisoner's dilemma payoff amount (reward, temptation, punishment, or sucker's) to its energy budget. If ego is a WAC or WAD and its partner defected (i.e., its partner's strategy = 0), ego sets moving to 1, making itself eligible to run *move* later in the time step. Next, ego asks its _partner_ to update its energy by adding the relevant prisoner's dilemma payoff amount to its current energy level. If partner is a WAC or WAD and ego's strategy = 0, then the partner sets moving to 1 to retaliate in response to ego's defection. Both players are marked as having played the prisoner's dilemma during the current time step (that is, both set played-already=1), preventing them from playing the prisoner's dilemma more than once during the time step.

move
Every player with moving = 1 moves one patch to the right or one patch to the left (each direction is chosen with equal probability—.5 to the right and .5 to the left) of its current cell. If the player moves to the left, it subtracts 90 from its heading so as to face the direction it just moved. If the player moves to the right, it adds 90 to its heading so as to face the direction it just moved. After moving to the right or left, the player sets moving = 0 to ensure that it cannot move more than once per time step.

reproduce
Players reproduce asexually. Players with energy >= 100 are asked to run *reproduce*. First, the parent halves its energy. Next, the parent compares a random floating point value between 0 and 1 to mu. If the random value is less than mu, the parent hatches a new player (an offspring) that adopts a breed drawn randomly from the parent's potential-breeds-list. If the random value is greater than mu, then the parent hatches a new player (an offspring) with the same breed as the parent. At birth, the offspring's energy is equivalent to its parent's energy, which was halved at the start of the *reproduce* submodel. There are two different types of offspring dispersal. Offspring can be placed randomly throughout the entire lattice (offspring-dispersal = global) or randomly within a local neighborhood around the parent (offspring-dispersal = local), where the size of the local neighborhood is defined by dispersal-radius.

enforce-carrying-capacity
This submodel runs for as long as the number of players in the population (after play and reproduction are completed) exceeds the carrying-capacity of the environment. The observer randomly selects one player and then subtracts ten units from its energy. If the selected player's energy is less than or equal to zero as a result of this reduction, it dies and is removed from the simulation. If the population size remains greater than carrying-capacity after this player has been selected and its energy reduced, the entire process is repeated (i.e., a player is randomly chosen from the population and its energy is reduced by 10 units). This procedure continues iteratively until enough players have died to return the population size to carrying-capacity. Although all players have an equal probability of being selected during each iteration of this

submodel, players with lower energy values are on average more likely to die due to density dependent mortality than players with higher energy values.

References

Aktipis, C. A. (2004) Know when to walk away: contingent movement and the evolution of cooperation. *Journal of Theoretical Biology* 231(2):249-260.

Grimm, V., et al. (2010) The ODD protocol: A review and first update. *Ecological Modelling* 221:2760-2768.

Premo, L. S. and J. R. Brown (2019) The opportunity cost of walking away in the spatial iterated prisoner's dilemma. *Theoretical Population Biology*.

Contact Information

This model was programmed by Luke Premo (Department of Anthropology, Washington State University and the Department of Human Evolution, Max Planck Institute for Evolutionary Anthropology) with conceptual contributions by Justin Brown (Department of Anthropology, Washington State University).

One of the best things about computer simulation programs is that they can often be improved with the help of other researchers. The more sets of eyes that pass over the code, the better. So, please, have a look at the source code under the "info" tab. Any comments, questions, or corrections are always welcome. You can contact me at:

Luke Premo
Department of Anthropology
Washington State University
Pullman, WA 99164-4910
luke.premo at wsu dot edu