ODD for "An agent-based model of hierarchical information-sharing organizations in asynchronous environments"

The model description follows the ODD (Overview, Design concepts, Details) protocol for describing individual- and agent-based models (Grimm et al. 2006; 2010; Railsback and Grimm 2018).

1. Purpose and patterns

The basic research question was whether hierarchies were useful in changing environments, and if so, when, how, and why?

We sought to create a model wherein a simple hierarchical organization attempts to adapt to a changing environment, and where that organization consists of workers and managers that see information at different scales: ground-level vs higher-level.

The model had both core requirements and constraints. Adaptation would require very simple learning on the part of both the workers and the managers. The environment should be as simple as possible while still being able to vary by worker location. Agents could have no ability to punish one another; we needed to eliminate social power. Thus, the organizational structure consisted only of communication between workers and managers.

The model achieved these basic components and patterns simply, although a simpler model may be possible.

2. Entities, state variables, and scales

The model has local environments, agents, and basic communication.

Environment:

We create a grid-cell environment that is simple while still making it possible for agents to experience different local environments in space and time. We first create a strip of cells where each one worker stands on one cell. Each worker's cell is their local state: that state is one of two binary values, and can change its value on each time step of the model. Static environments are of no interest, so the simplest environment in our set is a synchronous environment where all local states are the same in space and change their values simultaneously over time. These environments have little complexity, since every lane is a duplicate of its neighbor lanes. To create more complex environments, we modify the landscape by delaying or inverting the onset of environmental changes the agents experience relative to one another. In Figure 1 we visualize the one-dimensional world agents experience over time as two-dimensional landscapes, and show how simple algorithmic landscape modifications result in local conditions becoming asynchronous with one another.

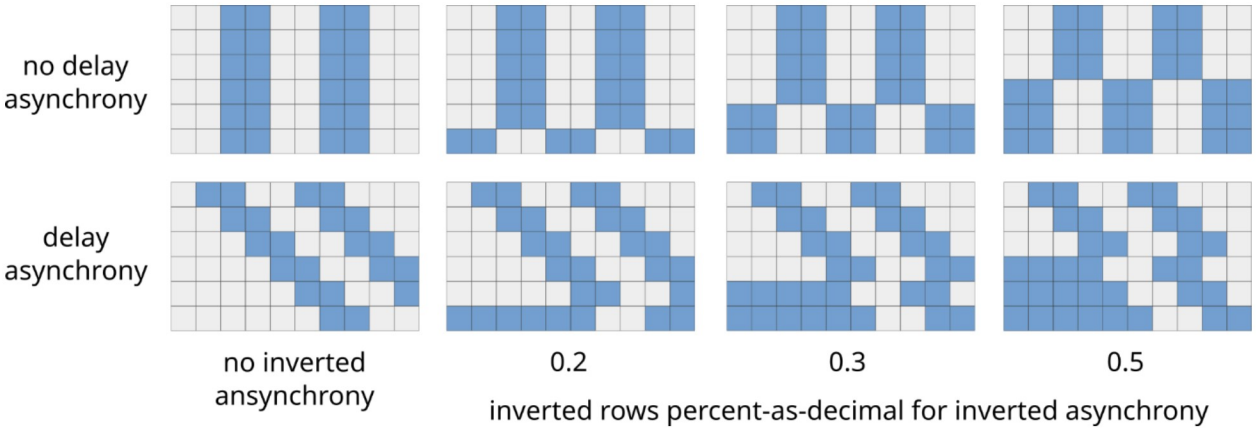

Figure 1. X-axis: inverted lanes, percent-as-decimal. Y-axis: delay asynchrony off or on. Each pattern results from a synchronous starting pattern transformed through a combination of these parameter settings. This selection of small submatrices from the simulated landscapes shows the differences in asynchrony. The set of environmental landscapes we use in the model are larger, and each is much longer than shown here. Within each pattern, each agent has its own row and each time step is a column; each of these patterns would have 6 workers, one on each of the 6 rows.

Each cell "problem" is a 0 or 1; each worker has its own lane of problems. When landscapes are synchronous, problem transitions happen across lanes simultaneously. When landscapes are asynchronous, that asynchrony is controlled by two parameters: 1) whether problems are staggered (delayed in transition) among the lanes, so that the transition hits each worker at a different time; 2) what percentage of lanes are made the inverse of other lanes. Staggering transitions among workers is a milder form of asynchrony, whereas inverted rows represent very different local sub-environments from the overall environment. By varying these parameters we can create increasingly asynchronous environments with either delayed environmental changes across agents, or some local conditions that are the inverse of the overall environment, or both.

Agents:

In this model, each agent must solve one problem per round by correctly predicting the incoming local environmental state. Agents–both workers and managers–have no ability to foresee future problems, so they must rely on their memory of past problems and on input from others. Agents have memories which store the last n problems they have seen (their memories are 3 slots to 9 slots long). When deciding on a strategy for the next problem, they consult their memory and any inputs (i.e., advice from the manager) and take the mode of that set of values. Agent capabilities and preferences are homogeneous within runs, but become varied in their memories and choices as they learn their heterogeneous environment. The agents' decision-making is focused only on getting the next tick right.

Agents have input weights that are a 0 to 1 ratio of how much weight their put on their memory vs their manager's input (assuming they're in the manager condition). Since the choice is the statistical mode, the inputs are multiplied in number (not in value) to give the mode a set

of inputs that are the correct ratio for that weight; in other words, the size of the set is scaled to be larger so that the number of elements is the right proportion. Agents also have weight adjustment increment settings, which at 0 mean that they cannot adjust the input weight value, but above 0 means they can adjust the weight by that value each tick (e.g., 0.2 means that if the agent decides to adjust the ratio, they will change it by 0.2 only, not more or less).

2.1. Variables and their parameters

Table 1 contains the key parameters for understanding the model. Simulations included every combination of these parameters.

2.1.1. Key parameter inputs for the environment

Parameter	Dynamic?	Possible Values	Description
number-of-managers	No	{0, 1}	no-hierarchy vs hierarchy
delayed-asynchrony	No	{0, 1}	Whether to delay environmental changes by 1 additional unit for each subsequent lane; this leads to a stair-step pattern in the environment in time and space.
inverted-asynchrony	No	{0, 0.1, 0.2, 0.3, 0.4, 0.5}	Percent-as-decimal of total lanes where all values are inverted.

2.1.2. Key data objects for the environment

Object	Dynamic?	Possible Values	Description
problem-matrix	No	{0, 1}	Stores all problems in the grid as a matrix.
turtle-adj-matrix	No	{0, 1}	Stores which turtles can communicate with one another as an adjacency matrix.

2.1.3. Key parameter inputs for the agents

Parameter	Dynamic?	Possible Values	Description
agent-mem-length	No	{3, 4, 5, 6, 7, 8, 9}	Number of past problem values each agent can store.
weight-others-input	Yes	{0, 0.25, 0.5, 0.75}	Weight agent puts on advice from other agents versus their own memory. A weight of 0 is

			equivalent to ignoring advice. If agents can adjust the weight, this variable is the starting weight.
weight-adj-increment	No	{0, 0.1, 0.2, 0.3, 0.4, 0.5}	How much an agent can adjust their input weight in one tick. If the value is 0, the agent does not adjust their input weights.

2.1.4. Key variables for the agents

Variable	Dynamic?	Possible Values	Description
my-weight-others-input	Yes/No depending on weight-adj-increment	[0, 1]	The weight, as a percentage, that the agent puts on advice from the manager versus their own memory. For example, a weight of .25 means the advice is weighed .25 and their memory is weighed .75 when making a decision. A weight of 0 is equivalent to ignoring advice. Runs in which agents have weigh-adj-increment set to above 0 implies the agent can (and probably will) be changing this value over the run.
my-mem	Yes	[0, 1]	The last n problems the agent remembers, where n is the memory length.
my-input-from-others	Yes	[0, 1]	The agent remembers what the manager told them so that they can judge whether the input was correct or not.
info-to-transfer	Yes	[0, 1]	The agent tells the manager what the answer to the last problem was.

3. Process overview and scheduling

3.1. Overview of the model routine

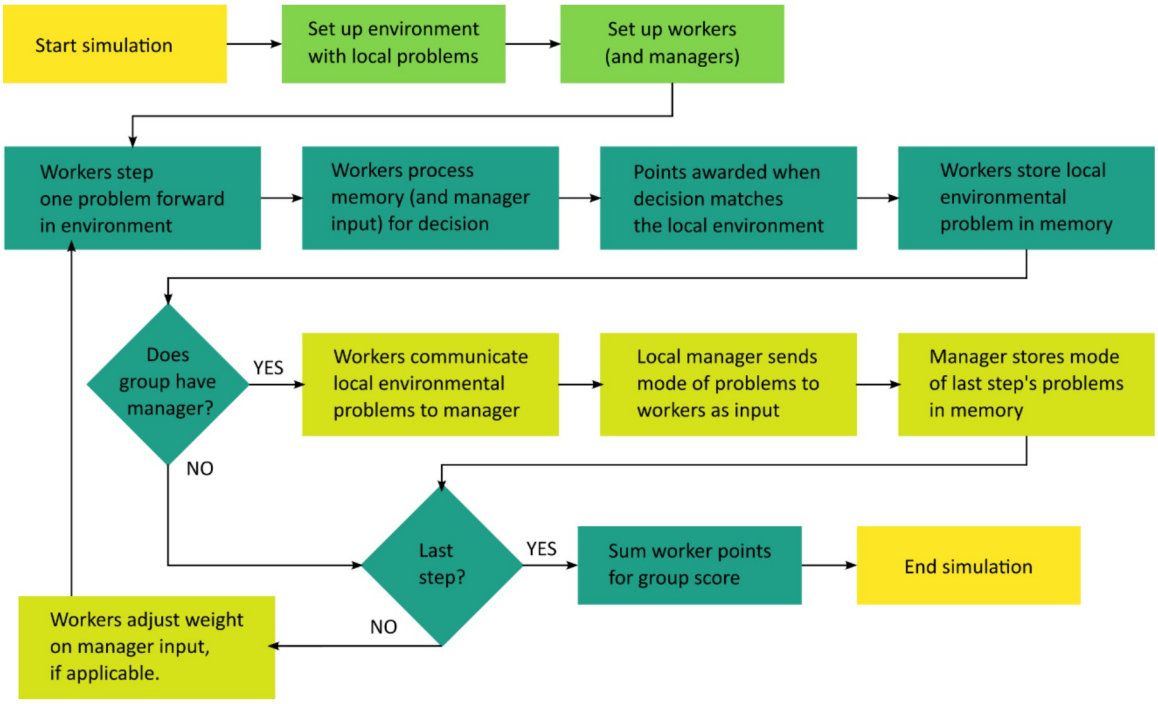

Figure 2. A flowchart of the model routine. The third row of boxes, and the box in the lower left, contain manager-specific processes that only happen in groups with managers. Note that managers never provide input on the current timestep to the workers, but only the previous timestep. Each worker is awarded points for successful decisions individually each timestep, but the individual scores are summed together when the simulation is complete; managers cannot earn points.

3.1. Setup/Initialization

Set-up environment

1. Set-up the problem matrix/grid according to the environmental input parameters.

Set-up agents

1. Create a worker class, and, if the run has a manager, a manager class.
2. Create the number of workers and managers specified in the input parameters. In the model used in the paper, the number of workers was always ten, with only one manager for hierarchical groups, and no manager for groups without hierarchy.

3.2. Run loop

1. If and only if the run has a manager

    a. Each worker communicates their last problem to the manager.
    b. The manager communicates the statistical mode of the last problem to the workers as input.
2. Each worker uses its memory and manager input (if enabled for the run), weighed according to its current variable value, to calculate the statistical mode. This is its prediction for the current problem.
3. The manager (if enabled for the run) uses its memory and worker input, weighed according to their current variable value, to calculate the statistical mode. This is its prediction of the current problem (which it will communicate to the workers the next time step).
4. Each worker and the manager store their prediction in their memory.
5. The true value of the current problem is revealed to each worker, who stores it in memory. The true statistical mode of the current problem is revealed to the manager (if the run has a manager).
6. Each worker and the manager (if enabled) scores whether their prediction for this time step was correct.
7. Each worker updates their score with whether they were correct. The manager does not contribute to the score.
8. If future problems remain on the grid, agents move forward to the next time step and set of problems; loop to Run step 1.

3.3. Group scoring

1. When the run has finished (there are no future problems remaining), do not count scores of early time steps, since the agents' memories will not be filled. The first n problems, where n = 10 + the agents' memory length, should not be counted in the score.
2. Sum the scores for all workers in the group to calculate the group score.

3.4. Conceptual diagram of the model process

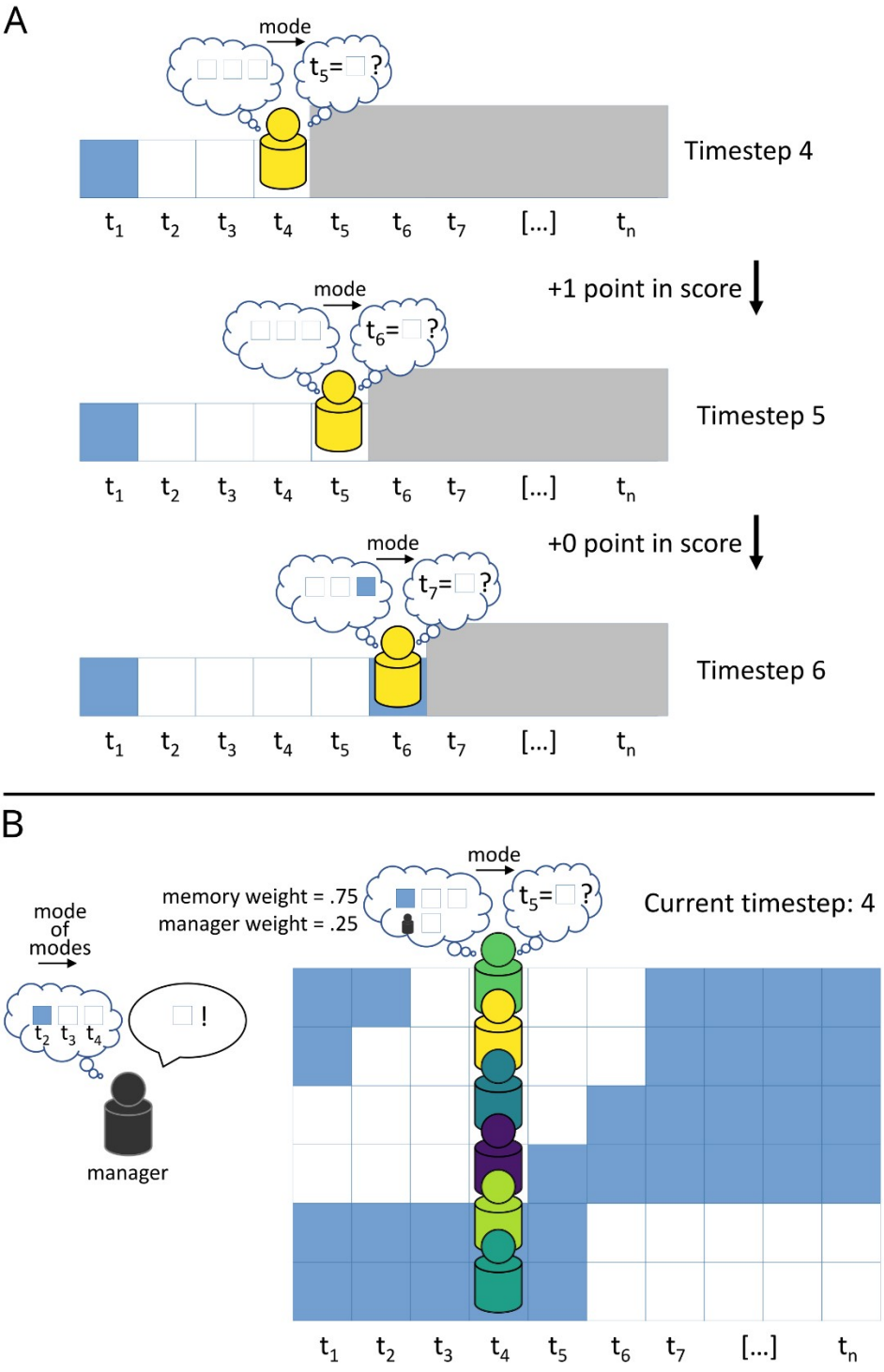

Figure 3. The core elements of agent decision-making about the environment in the model. Box A: A worker uses its memory of past timesteps to predict the future timestep, scoring a point when it is correct and scoring no points when it is not. Box B: A worker weighs manager input and its memory when guessing the next timestep. NOTE: The agents cannot see the cells in future timesteps; these have been revealed for reader convenience in box B.

4. Design concepts

This model is designed with the overarching concept of workers occupying a series of different locations, and these locations then change each timestep (tick in NetLogo). One can think of this as a board that agents advance on one timestep at a time. Alternatively, one can think of this like a treadmill with squares on the belt. We line workers up the width of a treadmill and have them float above the belt, each looking down at their cell, which is their local environment. When the treadmill advances one square, that is one unit of time passing (a timestep), and the workers see the next timestep's local environment. Thus, space and time can be represented by this 2D strip of squares, see Figure 1.

In the implementation, we place all worker agents on their own locale or cell (or patch in Netlogo). Managers, when active, do not have their own cell and do not interact with any spatial location, only with workers.

The primary design focus for the individual was an agent that could perform basic learning of the environment by predicting what the future state would be based on the past states. We chose to do this using the statistical mode of past states performed on the memory length of the agent (3 to 9 units in length). We also enable the agent to learn from additional information inputs, which in this study is manager input about the state of the environment.

The primary design focus for the social group / organization was the presence of a manager who could provide feedback about the overall state of the environment, which could be lossy—and was lossy any time the locales weren't all one value—because the environment can vary at each locale. We can then contrast how well workers do in different environments when they receive manager input on the overall state of the environment (hierarchy) versus not (no hierarchy).

4.1 Local Knowledge

Workers have local knowledge of their locale (local environment) stored in memory, and also can see the state of the locale at the end of each timestep, which they store in memory bank. Their memory bank operates on a first-in-first-out basis, where the oldest memory is dropped when the new memory is added.

The managers operate the same way, but their inputs are not the local environments but what the workers tell them the local environments are, which the managers then process (taking the statistical mode) to store as a single memory.

4.2. Organizational memory

The workers in the no hierarchy condition have memories of the past environmental states, but since they do not communicate it is not an organizational memory. However, workers in

hierarchy conditions with managers do have something approaching an organizational memory, consisting of their own memories and the manager's memory of the overall states of the environment. The parameter that allows agents to adjust the weight they put on the manager's input versus their own memories is a further element of organizational memory, as it effects the strength, in some sense, of elements of the memory.

4.3 Communication

Workers do not communicate with one another, but they do communicate with the manager (if present), and the manager communicates to them. These communications are inputs into their decision-making. These communications are *only* local environmental state, in the case of the worker, or the statistical mode of the past environmental state, if the manager. No other information is communicated.

When the group has a manager, at the start of each timestep, each worker communicates their last problem to the manager. The manager communicates the statistical mode of the last problem to the workers as input.

4.4 Autonomy

Workers that can adjust the input weight given to manager input are said to have some degree of autonomy. Workers who cannot are said to have no autonomy. Thus, whether the parameter for adjusting input weights is on determines whether the workers have autonomy in the model. If workers give only 50% weight to the manager, but cannot adjust that weight, they are said in our model to have no autonomy.

4.5 Changing environments

The environment can be thought of as local environments that change over time, one for each worker, and thus one lane for each worker. The overall environment is a tape or track of all these lanes together. Each timestep, local environments can change states between one of two binary values. These are determined ahead of running the agent part of the simulation by modifying a synchronous landscape. A synchronous landscape has all values change at the same time across all locales. This can be thought of as a perpendicular strip of one color on a longer tape or track of timesteps. Each landscape has multiple environmental changes over time.

Modifying this landscape is done through applying transformations as the parameter settings specify. First, if the parameter for inverted asynchrony is set to a value (e.g., .5), that value is interpreted as percent-as-decimal, and so that many worker lanes are changed so that each local value is switched to the opposite value in each of those lanes. Following that, if the parameter for delay asynchrony is set (only on or off), each lane is sequentially moved one timestep forward from the one before it, starting with the 2nd lane. This results in a stair-step pattern when the entire landscape is viewed (see Figure 1), but is seen by the group as environmental change where some local environments change before others.

4.6 Stochasticity

The model has a small amount of stochasticity. When deciding on a strategy for the next problem, they consult their memory and any inputs (i.e., advice from the manager) and take the statistical mode of that set of values; when two modes exist, agents pick one at random, which adds a small degree of stochasticity to the model.

4.7 Sensing

Agents sense their local environment perfectly. No part of the system has noise or error in sensing. This also includes the manager's received and given input to the workers.

4.8 Learning

Agent learning happens through storing past environmental values in memory. For the workers, these are the local values they have seen, and for the manager, these are the statistical modes of the local values they have been told by workers.

4.9 Prediction

Agents predict the future environment through processing their memories and (if the parameter is set) receiving manager input. If the parameter is set, workers weigh manager input by the set value (e.g., .5), and if the adjustment weight parameter is set, they can also adjust the weight they put on manager input the next time.

The adjustment is only performed when the manager's input was a better predictor than the worker's memory (manager is then weighted more then next timestep) or when the worker's memory was a better predictor than the manager's input (the manager's input is then weighted less the next timestep). Otherwise, they do not adjust weights from their current settings.

In other words, how much workers weigh their manager's advice is manipulated either as a static value across a run—non-adjustable by agents—or as a value which agents can adjust. When adjustable, workers assess their problem response each round and can adjust the weights incrementally. Workers decide to adjust weights only when their memory and the manager input disagree. If their memory was the better choice, they up-weigh memory. If listening to the manager was the better choice, they up-weigh manager input. If both were right or both were wrong, they leave the weights unchanged.

Workers that adjust weights use the following algorithm. Let us define a number counting function $\mathcal{C}(x)$ such that each of its components $i \in \{0, 1\}$ counts the respective number in a 0-indexed vector of binary values

$$\mathcal{C}(x)_i \coloneqq \sum_{j \in N \,|\, x_j = i} 1$$

For example,

$$\mathcal{C}([1 \quad 0 \quad 0 \quad 1 \quad 1]) = [2 \quad 3]$$

The counting function is applied to the worker memory vector and the manager input vector, and a weighted sum of the two is used to select the predicted environmental state x^* by picking the component with the highest value

$$x^* = arg\,max_{i\in\{0,1\}}\, w\mathcal{C}(x_w) + (1-w)\mathcal{C}(x_m)$$

Where x^* is the predicted environmental state, w is a scalar weight, and x_w and x_m are worker and manager input, respectively. In other words, the weighing algorithm takes two (binary) vectors, one being the worker's memory and one being manager input, each which is a series of 0s and 1s. These are then encoded as counts in a 2-tuple. For example, the memory vector [0,0,1,1,0,0] would be encoded as [4,2], and the manager input vector [1] would be encoded as [0,1]. These encodings enable a weight adjustment to each based on the agent's weight parameter, after which the highest value is taken, and then the 0 or 1 (based on the index of the encoding) is selected. For example, say the worker's memory is weighed .2, in which case manager input is weighted .8. Using the previous examples, the calculation is $.2 * [4,2] + .8 * [0,1] = [.8,1.2]$; we select the higher value from $[.8,1.2]$, which is 1.2, which is the second position in the 2-tuple, and which is associated with binary value 1. The worker then predicts the environmental state as 1, which can be thought of as "on" or blue in Figure 1.

4.10 Interaction

Workers in the no hierarchy condition do not interact with one another. Workers in the hierarchy condition interact directly with the manger, and the manager interacts directly with them; in this sense the workers interact with each other indirectly in the hierarchy condition.

This is further detailed in "4.3 Communication."

Workers and managers do not interact with the environment in the sense of changing it. However, the workers do score their success in dealing with the environment based on whether they predicted the correct state.

4.11 Output datasets

Output datasets include one or more (depending on settings) group score results for a specific parameter configuration. These are used to make dyad comparisons where all parameter settings are the same except one has a manager (hierarchical) and one does not (not hierarchical). We used the behavior space extension in NetLogo to run each parameter set 20 times due to the slight stochasticity of the model.

5. Initialization

Upon initialization, the model sets up the problem matrix/grid according to the environmental input parameters shown in "2.1.1. Key parameter inputs for the environment." Initialization includes the construction of the landscape (see Figure 1) from the environmental parameters, starting with a simple simultaneous-transitions environment with 9 transitions (top-left example in Figure 1). If the parameters of the particular run specify environmental asynchrony, the landscape is modified accordingly while retaining the same 9 transitions in each lane.

Examples of these modified landscapes are shown in Figure 1. Environments are always 255 problems long and 10 problems wide (one lane for each of the 10 workers; Figure 1 shows fewer lanes and shorter landscapes for ease of visualization).

Name on Netlogo GUI control element	**Parameter**	**Value for initialization**	**Varied between runs?**
number-of-workers	number-of-workers	10	No
problems-per-worker	problems-per-worker	255	No
transitions	transitions	9	No
stagger	delayed-asynchrony	{0,1} set by Behaviorspace during parameter sweep	Yes, see "2.1.1. Key parameter inputs for the environment", reproduced below.
oppositizer	inverted-asynchrony	{0, 0.1, 0.2, 0.3, 0.4, 0.5} set by Behaviorspace during parameter sweep	Yes, see "2.1.1. Key parameter inputs for the environment", reproduced below.
number-of-managers	number-of-managers	{0,1} set by Behaviorspace during parameter sweep	Yes, see "2.1.1. Key parameter inputs for the environment", reproduced below.

5.1.1. Key parameter inputs for the environment

Parameter	Dynamic?	Possible Values	Description
number-of-managers	No	{0, 1}	no-hierarchy vs hierarchy
delayed-asynchrony	No	{0, 1}	Whether to delay environmental changes by 1 additional unit for each subsequent lane; this leads to a stair-step pattern in the environment in time and space.

inverted-asynchrony	No	{0, 0.1, 0.2, 0.3, 0.4, 0.5}	Percent-as-decimal of total lanes where all values are inverted.

See Figure 2 for a flowchart showing the initialization in the context of the model's routines.

Note: Workers fill their memories up with first steps on the landscape before they encounter transitions and before they are scored so that empty or partial memories do not interfere with their decision making, or affect score differences between runs.

6. Input data

The model does not use input from external sources.

7. References

The template cites the following publications:

Grimm, V. and S. F. Railsback. 2005. *Individual-based modeling and ecology*. Princeton University Press, Princeton, New Jersey.

Grimm, V., U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. Heinz, G. Huse, A. Huth, J. U. Jepsen, C. Jørgensen, W. M. Mooij, B. Müller, G. Pe'er, C. Piou, S. F. Railsback, A. M. Robbins, M. M. Robbins, E. Rossmanith, N. Rüger, E. Strand, S. Souissi, R. A. Stillman, R. Vabø, U. Visser, and D. L. DeAngelis. 2006. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling* 198:115-296.

Grimm, V., U. Berger, D. L. DeAngelis, G. Polhill, J. Giske, and S. F. Railsback. 2010. The ODD protocol: a review and first update. *Ecological Modelling* 221:2760-2768. doi:10.1016/j.ecolmodel.2010.08.019.

Railsback, S. F. 2001. Concepts from complex adaptive systems as a framework for individual-based modelling. *Ecological Modelling* 139:47-62.

Railsback, S. F. and V. Grimm. 2018. *Agent-based and individual-based modeling: a practical introduction, 2nd edition*. Princeton University Press, Princeton, New Jersey.

Zurell, D., U. Berger, J. S. Cabral, F. Jeltsch, C. N. Meynard, T. Münkemüller, N. Nehrbass, J. Pagel, B. Reineking, B. Schröder, and V. Grimm. 2010. The virtual ecologist approach: simulating data and observers. *Oikos* 221:98-105. doi:10.1111/j.1600-0706.2009.18284.x