ADVANCED PROJECT I

Project Report

Supervisor: Prof. Dr. Agostino Merico

Contributor: Dr. Davi Tavares

Virtual Environment for Individual-Based Modeling

By Ralph Florent

May 29, 2019

Abstract

This report provides some facilities for understanding the virtual environment implemented to study the habitat use by waterbirds in coastal lagoons of the tropics. This virtual environment is based on an Agent-Based Modeling system using some assumptions that are derived from previous observations of waterbirds' behavior within some habitats in the tropics. Further, some detailed information is given about the carried-out tests and their results.

1 Introduction

Complex Systems is a field of science studying how individual components of a system give rise to collective behaviors and how the system interacts with its environment [1]. One of the approaches to studying a system interaction with its environment is through Agent-Based Modeling (ABM), a generalized framework for modeling and simulating dynamical systems.

In our case, we intend to study the habitat use by waterbirds in coastal lagoons of the tropics. Besides being a computational simulation, ABM is the closest modeling assumptions capable of providing a deeper understanding and interpretability of such a system.

This project report outlines the different steps to achieve a Virtual Environment (VE) using an ABM technique to characterize the waterbirds behaviors, adaptation, and evolution within a set of habitats with distinguishable properties. These steps are a reference to the Python code implementation of this VE, which serves as a demonstration basis to run and simulate the ABM. Finally, the results are analyzed and discussed under the algorithmic methods, the content structure, and the workflow scheme that are derived mostly from the typical traits of each component of the system.

2 Methodology

This section will explore the methods used to implement the core functionality of this project. This exploration includes the mention of the workflow scheme, the algorithm and content structure, and finally, the programmatically-implemented coding procedure.

2.1 Workflow Scheme

This project's workflow scheme consists of 3 main steps:

1. *Initialize*: stands for initial conditions
2. *Observe*: handles the graphical parts
3. *Update*: computes random movements based on the probability distribution of the corresponding factors.

where each step contains itself a series of internal subprocesses aiming a specific goal.
Important: *Observe in Figure 1 the remaining steps categorized as* Preconditions *and* Postconditions*. They represent respectively the* Before *and* After *the 3 main steps* ***`Initialize`**, **`Observe`***, and* ***`Update`*** *are executed. Note also that the* ***`Initialize`*** *process is considered part of the Preconditions semantics. That is because it only prepares the basic conditions for the components of the system, which are the habitats and the birds.*

Analyzing the workflow diagram in Figure 1, we denote the following fields:

- ***Start***: indicates the starting point of the VE simulation.
- ***Prior Considerations***: are the basic setup necessary to fulfill the initialization phase requirements. This setup spans the following elements: the geometry of the habitats and the human settlements; the functions defining the probability distribution of the random movements (driven by the water salinity, water depth, and food availability factors); the duration of the overall simulation process; and a reasonable threshold to handle the feasibility of the random movements for a given seabird under certain conditions.
- ***Initialize***: creates the initial conditions of the system based on prior considerations mentioned above. That is the patches (habitats) and agents (seabirds) creation.
- ***Observe***: generates a 2-dimensional plot whose scale goes from zero to one$(0-1)$ in both axes (x, y). The rendered plot helps to visualize both the patches' and agents' positions.

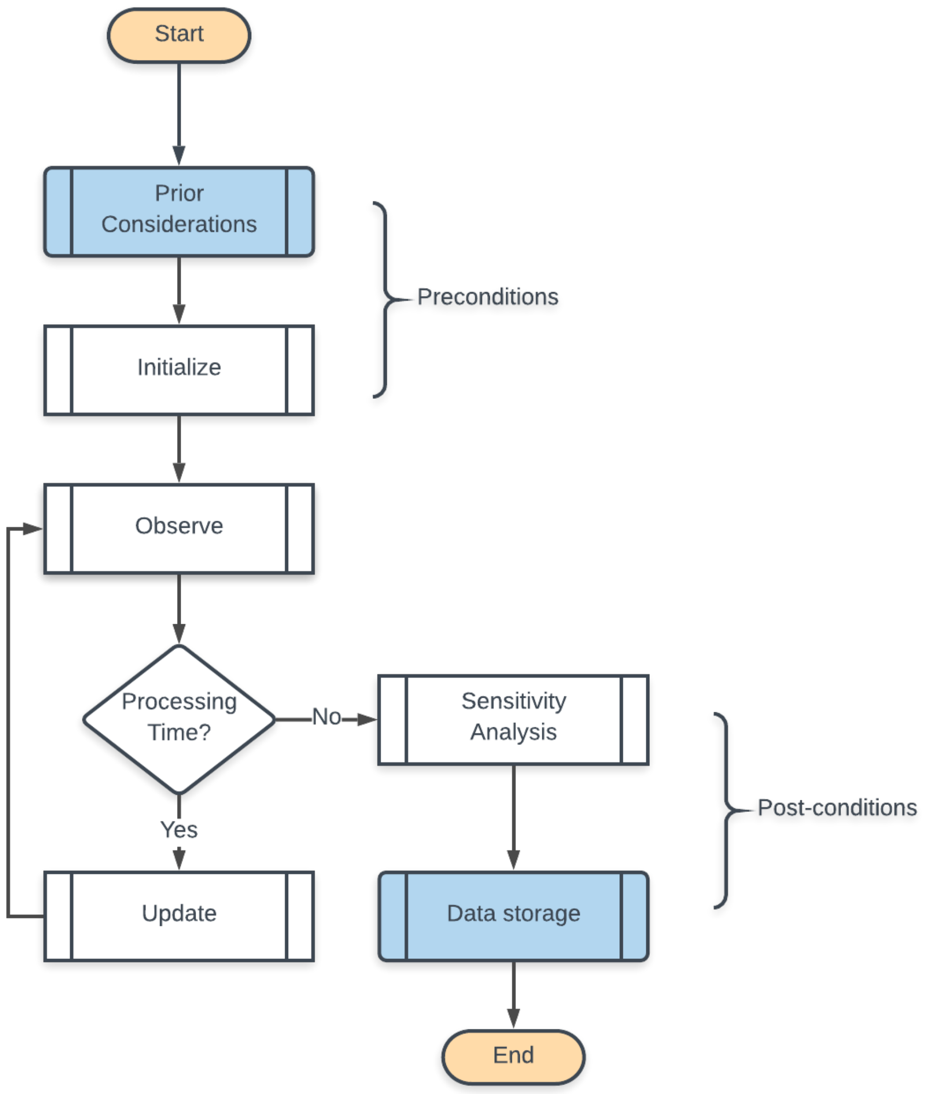


Figure 1: Workflow diagram
(credits: made with *Lucidchart*)

- ***Processing Time?***: focuses on updating the agents' positions' as long as the conditional parameter for the processing time holds. That is, the iteration is exclusively based on a specific number of times without accounting for other parameters that might influence the habitats and the birds. Note that, in this current version, the iteration is set statically during the prior considerations process.
- ***Update***: randomly assigns an agent to new positions within the existing habitats, considering a given threshold and the other aspects of the probability distribution.
- ***Sensitivity Analysis***: collects the probability values to form a set of probability distributions, which later can be analyzed and compared to each other with the expectation to draw conclusions on the final output.

- ***Data Storage***: given the generated plots, collects them as PNG images, and then generates a GIF [2] out of the entire dumped images. That is relevant to provide the end-user useful insights on the collected data.
- ***End***: indicates the ending point of the VE simulation.

Recalling that this Virtual Environment constitutes essentially a digital representation of an Agent-Based Modeling system, each component of such a system relies on the interaction and interconnection with other involved elements in an organized flow. Therefore, the diagram in Figure 1 shows a workflow scheme that intends to provide a visual aid for a better understanding of the system's behavior.

2.2 Algorithm & Data Structure

The VE simulation implies the use of well-coordinated processes and subprocesses, which, once computed, will eventually attempt to explain the agents' behavior and their mutual interactions with the environment in which they coexist. This section discusses the algorithm and data structure applied to construct these processes and subprocesses.

2.2.1 The *Habitat* and *Agent* data structure

In the VE simulation, both the wetland areas and the human settlements of the coastal lagoons are represented by the term *Habitat*[1], and the waterbirds, by the term *Agent*. In this case, the concept "Habitat" is a 2-dimensional *static* polygonal shape drawn from certain given geometrical measurements (see Figure 2). Similarly, the concept "Agent" is simply the representation of the waterbirds with some of its characteristics or attributes.

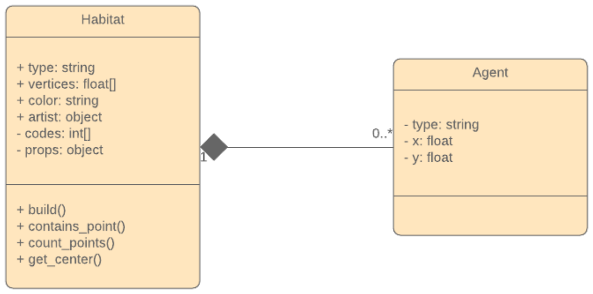


Figure 2: Data structure of *Habitat* and *Agent*
(credits: made with *Lucidchart*)

Observe that in Figure 2, we use the class diagram named *UML (Unified Modeling Language)* to model and document the properties of the components: Habitat and Agent. On the one hand, we construct a Habitat class definition with the following properties:

- ***type***: the type or category name of the habitat;
- ***vertices***: the coordinates of the patch representing the habitat;
- ***color***: the color (edge and face) to apply or distinguish a habitat from another;
- ***artist***: the patch-based polygonal shape to draw on a given figure;

[1]Note that human settlements are less appealing habitats for the waterbirds due to the humans' threatening characteristics

- ***props***: the dictionary-like additional properties that characterize this habitat;
- ***build()***: constructs the *artist* or patch on a figure;
- ***contains_point()***: determines whether or not an x-y coordinate (point) belongs to a patch;
- ***count_points()***: counts the total number of agents located within the patch based on their x-y positions.
- ***get_center()***: obtains the center point (x-y coordinate) of this habitat.

On the other hand, we construct an Agent class definition with the following properties:

- ***type***: the type or category name of the agent species;
- ***x***: the x-coordinate of the agent within an area;
- ***y***: the y-coordinate of the agent within an area.

Important: *Keep in mind that some of the methods in the class definitions use helper functions to do their specific task. These helpers can be found in the Python [3] scripts located in Appendix A.*

2.2.2 The overall algorithm

The overall algorithm is quite based on the step-by-step flow chart described in Figure 1. In other words, it corresponds to the descriptive, logical aspects of the core functionality of the VE. The steps are as follows:

1. ***Given***: given a collection of geometrical measurements (design) of the existing habitats and human settlements in a specific environment, a finite number (relatively small, 20 for example) of seabirds, and a set of predefined probability distribution functions (PDF) whose arguments are the characteristics of that environment;
2. ***Initialization***: represent digitally (virtually) that environment by creating patches and agents;
3. ***Update***: randomly choose an agent, then assess the probability of it moving to a random destination, and finally move the agent (if doable);
4. ***Observe***: snapshot the current state of the plotted environment, then save figure as a PNG image;
5. ***Iterate***: Repeat steps **3** & **4** for n times;
6. ***Stop***: collect the dumped images and form final GIF image to visualize the random movements of the agents.

2.2.3 The *Update* algorithm

Some of the processes are straightforward and do not demand a time-, or energy-consuming logic to build them. For instance, the initialization phase is one of the common cases where the developer only needs to take care of statically sets of values required as prior considerations for the initial conditions. But, as for the *Update* process, a rational, analytical solution is needed.

This algorithm defines an asynchronous approach to update an agent's status, namely its geolocation randomly. Thus, the set of instructions that follows below is the algorithm used to accomplish the "*Synchronous Update*" functionality of the VE simulation:

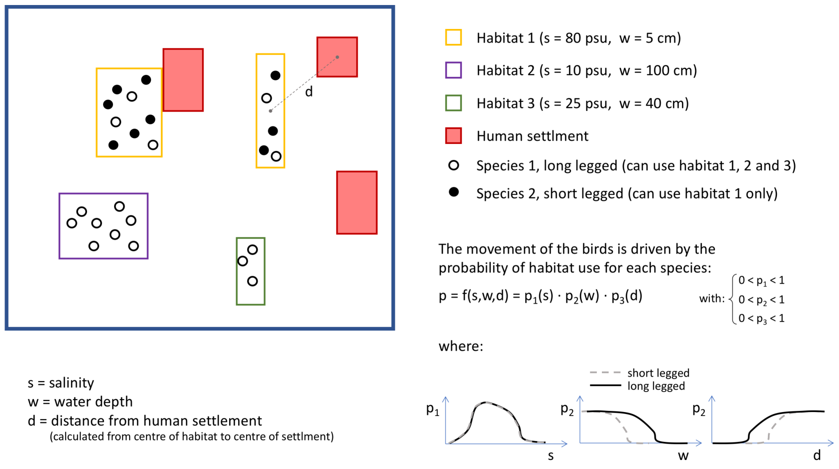
1. ***Given***: given a randomly selected agent;

2. ***Initialization***: randomly choose a new destination within an "acceptable" habitat (an area where this agent can move to, given the environmental conditions);
3. ***Computation***: compute the probability of that new destination use for this agent.
4. ***Update***: finally, move the selected agent to that new destination if the calculated probability complies with the threshold.

Recalling that this version of the project is a prototype whose purpose is to virtualize a static Agent-Based Modeling system, these algorithms are defined in their most simplistic model. For this reason, they are subject to change in the future when it comes to updating the dynamics of the system or adding more complex variations.

3 Results & Discussions

The virtual environment prototype is the end-result of 2 combined pilot tests following the general model concept illustrated in Figure 3. The results of these tests are presented in the subsections that follow.



d
Habitat 1 (s = 80 psu, w = 5 cm)
Habitat 2 (s = 10 psu, w = 100 cm)
Habitat 3 (s = 25 psu, w = 40 cm)
Human settlment
Species 1, long legged (can use habitat 1, 2 and 3)
Species 2, short legged (can use habitat 1 only)
The movement of the birds is driven by the probability of habitat use for each species:
p = f(s,w,d) = p1(s) · p2(w) · p3(d)
with: 0 < p1 < 1, 0 < p2 < 1, 0 < p3 < 1
where:
s = salinity
w = water depth
d = distance from human settlement
(calculated from centre of habitat to centre of settlment)
short legged
long legged
p1
s
p2
w
p2
d

Figure 3: The general model concept for studying habitat use by waterbirds in coastal lagoons of the tropics. The functions for calculating the probability of birds moving using a given water depth, distance from human settlement and salinity can be found in Appendix A.

3.1 First Pilot Test

In this very first test, we aim for the most possibly simplistic Agent-Based Modeling simulation [4]. The objective of this test is to simulate a physical environment of static habitats and a few randomly-positioned waterbirds with the expectations of moving around over time. In addition to that, we create a simple restriction rule, which is "*the waterbirds are not allowed to visit the habitats*".

Simulating this startup environment requires a focus on drawing specific patches (representing the habitats) and generating a finite number of agents (representing the waterbirds). In this case scenario, we use a square figure scaled from 0 to 1 in both sides as the limited area for the environment (see Figure 4). Then, with the help of the *matplotlib* [5, 6] 3rd-party library, we produce the patches [7]. Similarly, we use object-based definitions to create the agents and randomly position them within the environment using some helper functions — no prior considerations.

3.1.1 The patches

A patch is a drawing version of a habitat. It has a non-, or polygonal shape and tries to represent as closely as possible a 2-dimensional structure in reality. The shape can be very simple, as in the case of a simple square; and very complex, as in the case of a curvy closed-line. Besides its structural representation, a patch has some other properties such as *facecolor*, *edgecolor*, *lineweight*, and so forth, that are related to the setting of the drawing itself (see more details in [7]). But in our case, we choose to go with the rectangular shape, which we consider sufficient for the demo.

3.1.2 The agents

An agent is a drawing version of a waterbird. It can be of the type *short-legged* or *long-legged* species. This little nuance is what constitutes the core distinction in the waterbirds' behaviour within the habitats. Therefore, it remains relevant to set a clear cut in the design so that a short-legged type visually differs from a long-legged type.

Recall that the Agent class definition is really simple in terms of properties: *type and x-y positions*. The *type* attribute takes the value of either "short-legged" or "long-legged" and the agent's position is the x-y coordinates that take decimal values between zero (0) and one (1) within a 2-dimensional area. As a result, creating an agent in the VE simulation is to instantiate an object of the on-the-fly class Agent, then define its type as "short-legged" or "long-legged", and finally assign a random position to that agent. However, due to the restriction rule mentioned previously, the randomly-generated positions cannot fall into the area occupied by the habitats.

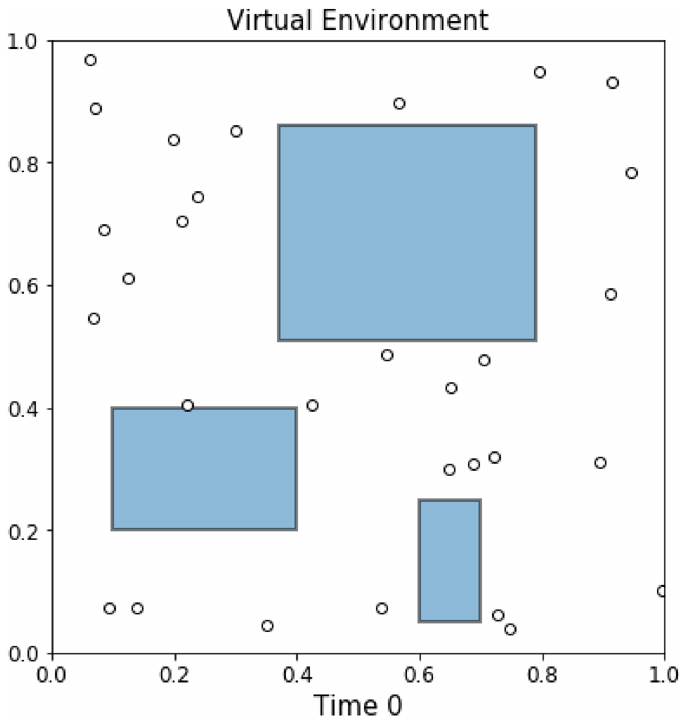


Figure 4: Preview of the first pilot test results: there is a total of 3 static patches (of different sizes) and 30 randomly-positioned short-legged agents. This visualization is the first generated plot of 20. The entire creation process is repeated 20 times, and a final GIF image is built up out of the 20 generated images for better visualization.

3.2 Second Pilot Test

In the first pilot test, we try to wrap up some key concepts and terminologies to avoid confusion with the interpretations of the end-results. The second pilot test, built on top of what is mainly explained in the first pilot test[2], adds a little bit more complexity to the simulation.

In the second pilot test, we focus on creating an approach that intends to bring the VE simulation closer to a real-life case scenario by randomizing the movements for the agents based on resource availability. That is, we clearly differentiate which patch an agent can move to by taking into account the maximum number of agents that can cohabit a patch at once. This particular denotation, resource availability, is the core principle used to determine whether or not a waterbird should move to another habitat to look for food since in real life the amount of waterbirds is directly proportional to food consumption. Obviously, other environmental aspects, such as the nature of the habitats are not being considered yet.

Another critical point to mention in this test is that it is related to the flowchart shown in Figure 1, excluding the *Sensitivity Analysis* step. Likewise, the test follows the general algorithm discussed in previous sections, except for the *Update* process. Consequently, this alters the implementation as well as the scripting.

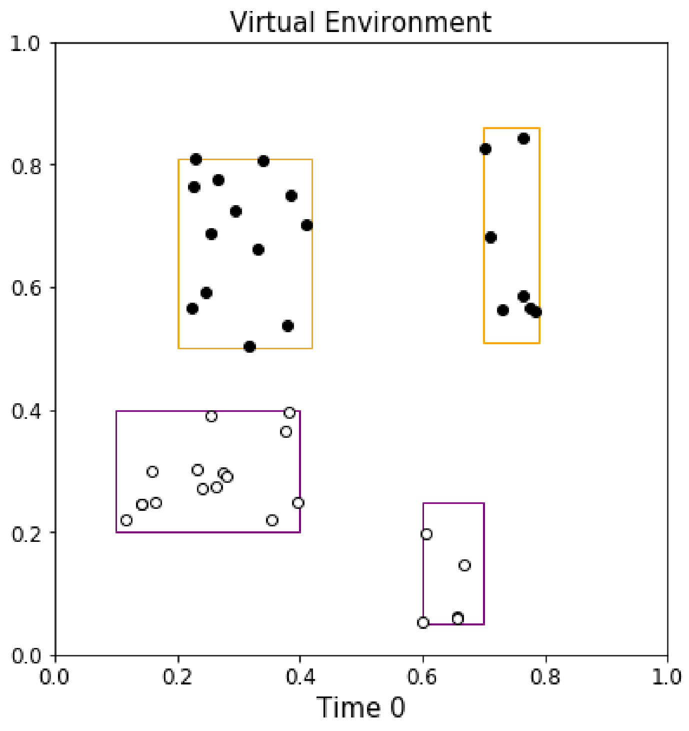


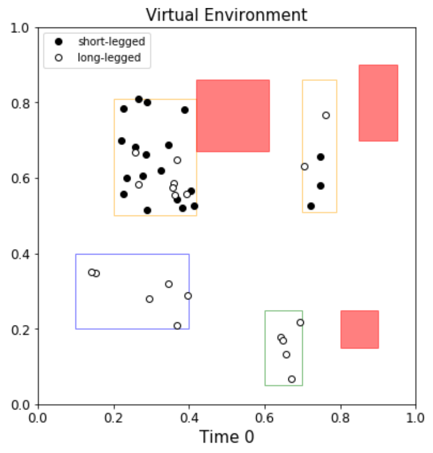
Figure 5: Preview of the second pilot test results: the yellow rectangles (patches) are the areas for long-legged waterbirds (agent) and the purple rectangles, the areas for the short-legged waterbirds. A long-legged agent can only travel between the yellow patches and a short-legged agent, between the purple patches. The initial conditions for the small rectangles (yellow and purple) are set according to the maximum capacity. Once the small patches reach the maximal capacity, an allowed agent can no longer travel across them unless the corresponding number of agents of the patch in question reduces over time.

[2]If not read yet, it is highly recommended to read the First Pilot Test part to grasp the whole idea of the second pilot test. Both are tightly coupled.

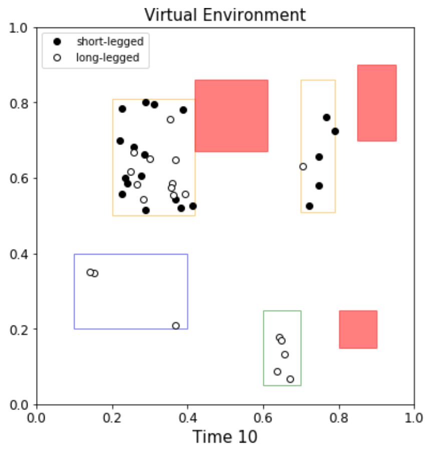
3.3 The VE Prototype

This part of the document discusses the current release of the ABM project, which is considered as the first prototype of the virtual environment simulation. The reason to state that is that this version goes beyond the first two pilot tests and covers important features of the VE system. It includes a higher degree of complexity, which brings the VE simulation much closer to the lifestyle of the waterbirds inhabiting the coastal lagoons of the tropics.

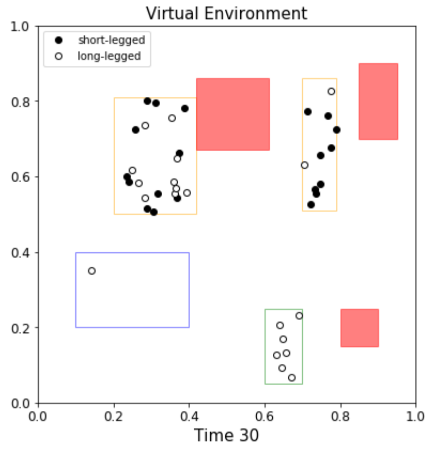
Again, this prototype relies on the terms and concepts clarified in the initial tests. Besides the other tests, this version follows all the programming principles, the algorithms, the workflow scheme, previously discussed in this document. It also uses the predefined probability distribution functions (PDF) for the environmental characteristics that play an essential role in the waterbirds' behavior within the habitats and their interactions with each other.



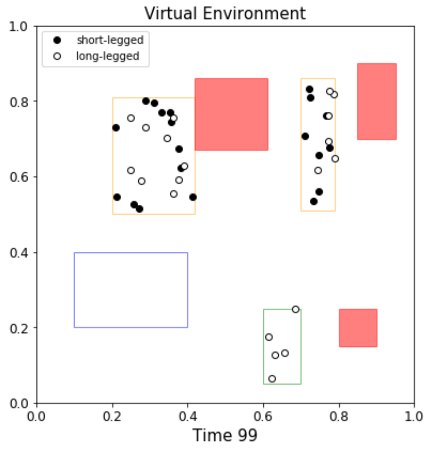
(a) Sample 1



(b) Sample 2



(c) Sample 3



(d) Sample 4

Figure 6: Preview of the VE simulation results: this visualization shows an extract of 4 samples out of 100 "processing random movements" units based on the PDFs. *Sample 1* and *Sample 4* are respectively the starting and ending point of the entire process. In one processing unit, only one agent gets to move, if the conditions are fulfilled to do so.

3.3.1 General comments

Unlike the other tests, the VE simulation setting of this prototype contains some updates, most importantly in the design. Firstly, there are more defined habitats: (edge-colored) `yellow, blue, green`, and (fully-colored) `pink` with some distinctive environmental characteristics each.

Next, the white dots and black dots are now labeled respectively as the long-legged waterbirds and short-legged waterbirds. Finally, the waterbird movements are driven by the habitat's category type. For instance, the long-legged waterbirds can move to any habitat except for the human settlements, whereas the short-legged waterbirds can only move to the *one* (`yellow`) habitats.

3.3.2 Analysis of the results

The results shown in Figure 6 are a single-run[3] of the scripts in Appendix A. We only extract four samples out of the 100 generated plots to explain the waterbirds' behavior and interactions under a specific initial condition (see Table 1). And this initial condition can only generate a single-point dataset for each PDF over time. To achieve a data set, we need to collect a set of initial conditions, especially for the habitats' characteristics and tune their values over time. For instance, simulating a water depth reduction or tweaking the food availability factors are an excellent example of how to achieve a multi-point dataset for each PDF. But since this is out of the scope of this release, we only mention it as our next considerations for future releases.

Initial Conditions	
Total of long-legged waterbirds	20
Total of short-legged waterbirds	20
Processing times	100
Threshold for	$1 * e^{-7}$
Areas for short-legged waterbirds	`one`
Areas for long-legged waterbirds	`one, two, three`
Habitat one (big)	s = 80psu, w = 5cm, f = 0.3
Habitat one (small)	s = 80psu, w = 5cm, f = 2.56
Habitat two	s = 10psu, w = 5cm, f = 6.41
Habitat three	s = 25psu, w = 40cm, f = 11.53

Table 1: Default values and parameters for the VE prototype's initial conditions. Note that the geometrical measurements of the habitats are part of the initialization process, but omitted here (refer to Appendix A for more details on them).

The first sample, *Time 0*, is the first generated plot with a random distribution of the waterbirds within the allowed areas. Note that the long-legged birds (white dots) are spread out in all four habitats whereas the short-legged birds (black dots) are only located in the habitats categorized as *one*. Then in the second sample, *Time 10*, we notice that the long-legged birds start leaving Habitat *two*. And finally, in the last 2 samples, Habitat *two* remains empty. The reason for this is that the movements of the long-legged birds in this particular case are reasonably driven by the PDFs and the chosen threshold. That is, the chosen threshold factor does not have a reasonable probability (50% each) of whether a bird should move or not.

[3]Running the script yourself will not achieve the same results due to the random initialization state.

4 Conclusion

This document reports the analysis and results of the Agent-Based Model when studying habitat use by waterbirds in coastal lagoons of the tropics. We have built a virtual environment that is computationally inexpensive by simply considering a few designing aspects to simulate the ABM under certain conditions. So far, the realization of this virtual environment is very promising and brings us closer to the real-life situation of the waterbirds' behavior.

The current version of the virtual environment is a prototype that has room for a lot of improvements. Throughout the document, some of these improvements are mentioned as well as some important points that are expected to be tackled in future releases.

References

[1] Agostino Merico. *Complex System and Agent-Based Modelling*. Jacobs University Bremen, Nov. 2018. (accessed: 29.05.2019).

[2] The Python community. *imageio*. Jan. 2019. URL: https://imageio.readthedocs.io/en/latest/userapi.html.

[3] Python Software Foundation. *Welcome to Python.org*. Mar. 2019. URL: https://www.python.org/.

[4] Project Jupyter. *Project Jupyter*. Jan. 2019. URL: https://jupyter.org/. (Last updated April 12, 2019).

[5] The Matplotlib development team. *matplotlib.path*. Mar. 2019. URL: https://matplotlib.org/3.1.0/api/path_api.html.

[6] The Matplotlib development team. *matplotlib.pyplot.plot*. Mar. 2019. URL: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html.

[7] The Matplotlib development team. *matplotlib.patches.Patch*. Mar. 2019. URL: https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.patches.Patch.html.

Appendix A Code Repository

All the code implemented and utilized during the execution of the simulation described in this report is available on the GitHub repository https://github.com/systemsecologygroup/BirdsABM.