Modelacion de Sistemas Multiagentes

- Diego Rosas A01634154
- Alejandro Pizarro A01633784

- Jueves 18 de Agosto de 2022

```
#%pip install agentpy seaborn

# Model design
from pickle import DICT
import agentpy as ap
import random

MOVES = [(1,0), (0,1), (-1, 0), (0,-1)]

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import IPython

class RoomModel(ap.Model):

    def setup(self):

        # Create agents (robots)
        self.robot = ap.AgentList(self, self.p['robot'])
        self.trash = ap.AgentList(self,
int(self.p['M']*self.p['N']*self.p['trash']))

        # Create grid (trash)
        self.room = ap.Grid(self, [self.p['M'], self.p['N']],
track_empty=True)
        self.room.add_agents(self.robot, [(1,1)]*self.p['robot'])
        self.room.add_agents(self.trash, random=True, empty=True)


        # Initiate a dynamic variable for all trash
        # Condition 0: Robot, 1: Basura, 2: Basura (clean)
        self.robot.type_agent = 0
        self.trash.type_agent = 1

    def step(self):

        robots = self.robot

        # Spread robot for cleaning
        for robot in robots:
            for neighbor in self.room.neighbors(robot):
                if neighbor.type_agent == 1:
```

```
                    neighbor.type_agent = 2
                    break
            else:
                self.room.move_by(robot, random.choice(MOVES))


        trash_pending = self.trash.select(self.trash.type_agent == 1)
        # Stop simulation if no dirty cells are left
        if len(trash_pending) == 0:
            self.stop()

    def end(self):

        # Document a measure at the end of the simulation
        cleaned_cells = len(self.trash.select(self.trash.type_agent ==
2))
        self.report('Percentage of trash',
                    cleaned_cells / len(self.trash))

# Define parameters

parameters = {
    'steps' : 250,
    'robot' : 25,
    'M' : 40,
    'N' : 40,
    'trash' : 0.3
}
sample = ap.Sample(parameters, n=30)

# Create single-run animation with custom colors

def animation_plot(model, ax):
    attr_grid = model.room.attr_grid('type_agent')
    color_dict = {0:'black', 1:'#d62c2c', 2:'#E7E6E6', None:'#E7E6E6'}
    ap.gridplot(attr_grid, ax=ax, color_dict=color_dict, convert=True)
    ax.set_title(f"Simulation of a robot cleaning\n"
                 f"Time-step: {model.t}, Trash pending: "
                 f"{len(model.trash.select(model.trash.type_agent ==
1))}")

fig, ax = plt.subplots()
model = RoomModel(parameters)
animation = ap.animate(model, fig, ax, animation_plot)
IPython.display.HTML(animation.to_jshtml(fps=15))
```

```
<IPython.core.display.HTML object>
```

```
# Perform experiment
exp = ap.Experiment(RoomModel, sample, iterations=40)
results = exp.run()
```

```
Scheduled runs: 40
Completed: 40, estimated time remaining: 0:00:00
Experiment finished
Run time: 0:00:06.483066
```

```
# Save and load data
results.save()
results = ap.DataDict.load('RoomModel')
```

```
Data saved to ap_output/RoomModel_3
Loading from directory ap_output/RoomModel_3/
Loading info.json - Successful
Loading parameters_constants.json - Successful
Loading parameters_log.json - Successful
Loading reporters.csv - Successful
```