Lab 3: El Farol Bar Problem[1]

Released: Monday 5th March, 2018
Deadline: March 9, 2018
Weight: 10 % for 06-27819, or 11.25 % for 06-27818 %

You need to implement one program that solves Exercise 1-3 using any programming language. In Exercise 5, you will run a set of experiments and describe the result using plots and a short discussion.

(In the following, replace `abc123` with your username.) You need to submit one zip file with the name `niso3-abc123.zip`. The zip file should contain one directory named `niso3-abc123` containing the following files:

- the source code for your program
- a Dockerfile (instructions in the appendix)
- a PDF file for Exercises 4 and 5

The El Farol Bar problem is a classical game-theoretic problem studied in economics. In Santa Fe, there is a bar called El Farol. Every Thursday night, the bar organises an event which everyone in the town wants to attend. If less than 60 % of the population attends the bar, all have a better time in the bar than staying home. However, if 60 % or more of the population attends the bar, it becomes too crowded, and it would have been better for everyone to stay at home. We assume that everyone decides individually whether to go to the bar without communicating with anyone. However, the attendance in the bar during all past weeks is known to everyone.

We will model the scenario using co-evolution. We will represent the strategies of each player with a state-based representation $S = (p, A, B)$, where $p = (p_1, \ldots, p_h)$ is a vector of "attendance" probabilities, $A = (a_{ij})$ is a state transition matrix in case the bar is crowded, and $B = (b_{ij})$ is a state transition matrix in case the bar is not crowded.

In any week t, the individual is in one of h states. If the individual is in state i in week t, then she goes to the bar with probability p_i that week. If the bar was crowded in week t, then in the following week, she transitions to state j with probability a_{ij}. If the bar was not crowded in week t, then she transitions to state j with probability b_{ij} in week $t + 1$. The individuals in the population all start in state 0, but have different strategies.

This hypothetical problem models many real-world economic scenarios. We are interested in understanding the conditions under which the population can achieve an efficient solution, i.e., a solution where the utilisation of a limited resource (in this case the bar) is as close as possible to its capacity. The problem is interesting, because the population cannot use the resource efficiently if all individuals use the same deterministic strategy.

[1] Revised: Monday 5th March, 2018 at 13:26.

Exercise 1. *(10 % of the marks)*

Implement a routine to sample from a distribution over the integers $\{0, \ldots, n-1\}$*, where the distribution is represented by a vector of* n *probabilities.*

Input arguments:

- `-prob` *vector of* n *probabilities*
- `-repetitions` *number of samples*

Output:

- *an element between 0 and* $n-1$

Example:

```
[pkl@phi ocamlec]$ app_niso_lab3 -question 1 -prob "0.25 0.25 0.25 0.25" -repetitions 5
2
1
3
2
0
```

Exercise 2. *(10 % of the marks) Implement a routine which simulates one step of a given strategy. Given the current state of the individual in week t, and whether the bar was crowded or not, you should return the state in week $t+1$ and the decision whether to go to the bar in week $t+1$.*

Input arguments:

- `-strategy` *a representation of a strategy as follows,*

$$h\ p_0\ a_{00}\ a_{01}\ \ldots\ a_{0h}\ b_{00}\ b_{01}\ \ldots\ b_{0h}\ p_1\ a_{10}\ \cdots,$$

  where h is the number of states in the representation.

- `-state` *the state in week t*
- `-crowded` *1 if the bar is crowded in week t, 0 otherwise*
- `-repetitions` *number of repetitions*

Output:

The output should be one line per repetition, where the following values are shown separated by the tab character

```
d s
```

where

- `d` *is the decision whether to go (1) to the bar or not (0) in week $t+1$*
- `s` *is the next state*

Example:

```
[pkl@phi ocamlec]$ app_niso_lab3 -question 2 \
        -strategy "2 0.1 0.0 1.0 1.0 0.0 1.0 0.9 0.1 0.9 0.1" \
        -state 0 -crowded 1 -repetitions 5
1 1
1 1
1 1
1 1
1 1
[pkl@phi ocamlec]$ app_niso_lab3 -question 2 \
        -strategy "2 0.1 0.0 1.0 1.0 0.0 1.0 0.9 0.1 0.9 0.1" \
        -state 1 -crowded 0 -repetitions 5
0 0
0 0
1 1
0 0
0 0
```

Exercise 3. *(40 % of the marks)*

Design a co-evolutionary algorithm for the El Farol Bar problem. Your algorithm should keep a population of strategies. The strategies should be evaluated via simulation over a fixed number of weeks. For each week, an individual receives payoff 1 if he attends the bar when it is not crowded (i.e., less than 60 % of the population in the bar), or a payoff of 1 if he is at home while the bar is overcrowded, otherwise the payoff is 0. The selection mechanism should favour individuals with higher payoff. You can use any of the standard variation operators, such as mutation and crossover.

Input arguments:

- `-lambda` *the population size*
- `-h` *the number of states in the strategies*
- `-weeks` *the number of weeks to simulate per generation*
- `-max_t` *the number of generations to run the simulation*

Output:

The output should be one line for each week, where the following values are shown separated by the tab character

```
tw tg b c a1 a2 ... alambda
```

where

- `tw` *is the week number*
- `tg` *is the generation number*
- `b` *is the number of individuals in the bar*
- `c` *is 1 if the bar is crowded and 0 otherwise*
- `ai` *is 1 if individual* $i \in [\lambda]$ *attended the bar and 0 otherwise*

Exercise 4. *(10 % of the marks)*

Describe your algorithm from Exercise 3 in the form of pseudo-code (see Lab 1 for an example). The pseudo-code should be sufficiently detailed to allow an exact re-implementation.

Exercise 5. *(30 % of the marks)*

In this final task, you should try to determine parameter settings of your co-evolutionary algorithm which leads the population to as efficient utilisation of the bar as possible (i.e., close to 60 %), while still not being overcrowded.

Your algorithm is likely to have several parameters, such as the population size, mutation rates, selection mechanism, and other mechanisms components, such as diversity mechanisms.

Choose parameters which you think are essential for the behaviour of your algorithm. Run a set of experiments to determine the impact of these parameters on the efficiency of the population. For each parameter setting, run 100 repetitions, and plot box plots of the average attendance over several weeks. Discuss and explain the results.

A. Docker Howto

Follow these steps exactly to build, test, save, and submit your Docker image. Please replace *abc123* in the text below with your username.

1. Install Docker CE on your machine from the following website:
   `https://www.docker.com/community-edition`

2. Copy the PDF file from Exercises 4 and 5 all required source files, and/or bytecode to an empty directory named niso3-abc123 (where you replace abc123 with your username).

```
mkdir niso3-abc123
cd niso3-abc123/
cp ../exercise.pdf .
cp ../abc123.py .
```

3. Create a text file `Dockerfile` file in the same directory, following the instructions below.

```
# Do not change the following line. It specifies the base image which
# will be downloaded when you build your image. We will release a new
# base image for each lab.

FROM pklehre/niso-lab3

# Add all the files you need for your submission into the Docker image,
# e.g. source code, Java bytecode, etc. In this example, we assume your
# program is the Python code in the file abc123.py. For simplicity, we
# copy the file to the /bin directory in the Docker image. You can add
# multiple files if needed.

ADD abc123.py /bin

# Install all the software required to run your code. The Docker image
# is derived from the Debian Linux distribution. You therefore need to
# use the apt-get package manager to install software. You can install
# e.g. java, python, ghc or whatever you need. You can also
# compile your code if needed.

# Note that Java and Python are already installed in the base image.

# RUN apt-get update
# RUN apt-get -y install python-numpy

# The final line specifies your username and how to start your program.
# Replace abc123 with your real username and python /bin/abc123.py
# with what is required to start your program.

CMD ["-username", "abc123", "-submission", "python /bin/abc123.py"]
```

4. Build the Docker image as shown below. The base image `pklehre/niso-lab3` will be downloaded from Docker Hub

```
docker build . -t niso3-abc123
```

5. Run the docker image to test that your program starts. A battery of test cases will be executed to check your solution.

```
docker run niso3-abc123
```

6. Once you are happy with your solution, compress the directory containing the Dockerfile as a zip-file. The directory should contain the source code, the Dockerfile, and the PDF file for Exercise 4 and 5. The name of the zip-file should be `niso3-abc123.zip` (again, replace the abc123 with your username).

7. Submit the zip file `niso3-abc123.zip` on Canvas.