Advanced use of the Flee code.

Written by Derek Groen (Derek.Groen@brunel.ac.uk), with help from Hamid Arabnejad, Diana Suleimenova and Gebremariam Assres.

In this 30-minute tutorial, you will learn how to run the parallel version of Flee and how to run a coupled simulation. The content of this tutorial is derived from two conference papers, which are - Groen, D., 2018, June. Development of a multiscale simulation approach for forced migration. In International Conference on Computational Science (pp. 869-875). Springer, Cham. - Groen et al., Towards Modelling the Effect of Evolving Violence on Forced Migration, accepted for the Winter Simulation Conference 2019

In this tutorial we will cover the following aspects:

- How to run the parallel Flee code using MPI.
- How to run a coupled simulation, with a microscale and macroscale migration model.
- How to run a coupled simulation, with Flee and a conflict evolution model

We will focus primarily on just running basic versions of each type, so that you get a rough idea how these things would work on your local machine.

Although this tutorial is brief, it may still be difficult to get everything working in 30 minutes, so feel free to pick your favourite subsection and start with that.

Requirements

To do this tutorial, you need a working Python3 installation, a working MPI installation, and the MPI4Py library. The numpy, pandas and matplotlib Python3 libraries are also recommended.

You also need to have Flee installed (http://www.github.com/djgroen/flee-release), as well as Flare (http://www.github.com/djgroen/flare-release).

Parallel Flee

The simplest way to run the parallel code is to run it with the *test_par.py* testing script. To run a parallel version of Flee, please:

1. Make sure you've navigated the the Flee installation directory in the terminal.
2. Type *mpirun -np 2 python3 test_par.py 5* to run Flee across 2 cores.
3. To measure the execution time of your run, type *time mpirun -np 2 python3 test_par.py 5* to run Flee across 2 cores.

Simple exercises

You can change *-np 2* to other values, such as *-np 1* to run on 1 core, or *-np 4* to run on 4 cores.

1. Use the aforementioned time command (see step 3 above) to time 10 runs of Flee using 2 cores. What's the highest time you get? And the lowest?
2. Use the aforementioned time command (see step 3 above) to a run of Flee using 1, 2 and 4 cores. Which core count gives you the lowest execution time?
3. (advanced) Open the test_par.py script. Are you able to change the number of agents in the simulation by editing this script? If so, how does changing that number affect your performance and scalability?

Flee + a microscale model

To run a basic micro-macro coupled model, you can use the test script *mscalecity.py*. Because you are running two models, you'll need to open two terminals for this part.

1. In the first terminal, go to your Flee installation directory, and type *python3 mscalecity.py 0*. This will start a microscale model of a small town.
2. In the second terminal, go to your Flee installation directory, and type *python3 mscalecity.py 1*. This will start a macroscale model of a hypothetical country.

Once you have typed the second command, you should see output being written to the screen for both simulations. You can write output to a CSV file, by appending > *out.csv* to the command described in step 2.

Flee + conflict evolution

A detailed tutorial on running Flee with conflict evolution, using the FabSim3 automation toolkit, can be found here: https://github.com/djgroen/FabFlee/blob/master/doc/Tutorial.md . In this simplified tutorial, we simply explain the manual steps you can take to run Flare, transport the output data, and run a Flee simulation based on the obtained conflict evolution.

To do this tutorial, you will need to download all the files in the following directory: https://github.com/djgroen/FabFlee/tree/master/config_files/mali . You also need to have the Flee and Flare codes installed.

To run Flare easily, you should make a script called *run_flare.py*, which should contain the following code:

```
from flee import InputGeography
from flare import Ecosystem
import numpy as np
import sys

if __name__ == "__main__":

  end_time = 100

  if len(sys.argv)>1:
      if (sys.argv[1]).isnumeric():
          end_time = int(sys.argv[1])

  if len(sys.argv)>2:
      input_dir = sys.argv[2]
  else:
      input_dir = "test_input_csv"


  if len(sys.argv)>3:
      out_file = sys.argv[3]
  else:
      out_file = "flare-out.csv"


  ig = InputGeography.InputGeography()

  ig.ReadLocationsFromCSV("%s/locations.csv" % input_dir)

  ig.ReadLinksFromCSV("%s/routes.csv" % input_dir)
```

```
e = Ecosystem.Ecosystem()

lm = e.StoreInputGeographyInEcosystem(ig)

#print("Network data loaded")

file = open("%s" % out_file,"w")

output_header_string = "#Day,"

for l in e.locations:
    output_header_string += " %s," % (l.name)

output_header_string += "\n"
file.write(output_header_string)

for t in range(0,end_time):

    e.evolve()

    output = "%s" % t

    for l in e.locations:
        if l.flare:
            output +=",1"
        else:
            output +=",0"

    output += "\n"
    file.write(output)

file.close()
```

For convenience, place this file in the same directory where you have placed the input files for the Mali conflict.

1. To run Flare, you can then type *python3 run_flare.py 300 input_csv input_csv/conflicts.csv*. This will generate a new *conflicts.csv* file, which you can load into Flee.
2. To run Flee, stay within the same directory, and type *python3 run.py input_csv source_data 300 > out.csv*.
3. To visualize the result, you can use the *out.csv* file with your plotting scripts as you have done before in the Flee tutorial.

Advanced/Optional material

If you are interested in incorporating weather data with the Flee model (or your own), please drop us a line, as we are currently working on that topic.

If you would like to learn more about how to use the model with supercomputers, and about tools to do uncertainty quantification, please have a look here: https://www.vecma-toolkit.eu/tutorials/ . Note that this page also contains tutorials on CFD models, Fusion-related models, and molecular dynamics models.

A good general reference on agent-based simulation in general can be found here:

- C M Macal (2016) “Everything you need to know about agent-based modelling and simulation” Journal of Simulation 10(2) p. 144-15

And lastly, if you wish to try out other agent-based modelling platforms, have a look at:

- NetLogo: https://ccl.northwestern.edu/netlogo/docs/ or https://netlogoweb.org/ (web-based version)
- RePast: https://repast.github.io/docs.html

Lastly, we maintain a GitHub repository giving information about nearly anything that we could think of in academia right here: http://www.github.com/djgroen/student-resources .