Bio4CompSimpy User Manual

Goals:

There are two goals to this manual

1. Helping end users understand how to interact with the system and create simulations for different Bio4Comp biological devices

2. Explaining the code structure to programmers who wish to expend it/ improve it.

Installation:

Download the Bio4CompSimpy folder from our GitHub repository: <https://github.com/ItamarDayan/Bio4CompSimpy>, and place it inside your beacon-calculus folder.

\* The program was written for use on Unix based operating systems only

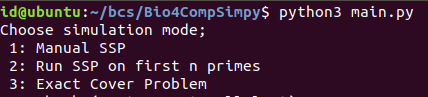
1. END USER MANUAL:

To start Bio4CompSimpy navigate in the terminal to the working directory of Bio4CompSimpy and then run main.py

(using the following command: python3 main.py).

Note that the code was written in python3 so running it on older versions probably will not work.

Running main.py will show you the main menu:



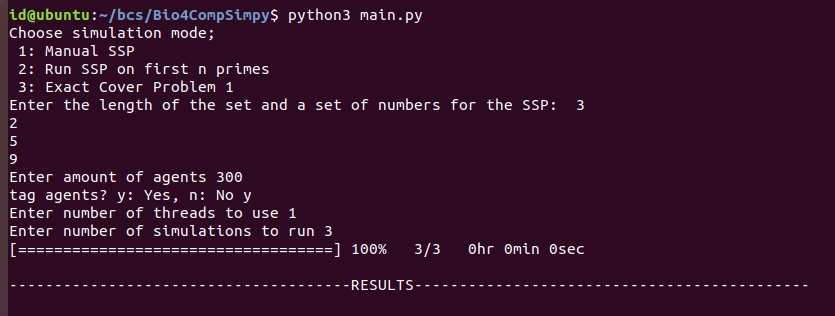
Here you can choose the problem you want to solve using the B4CMP simulator

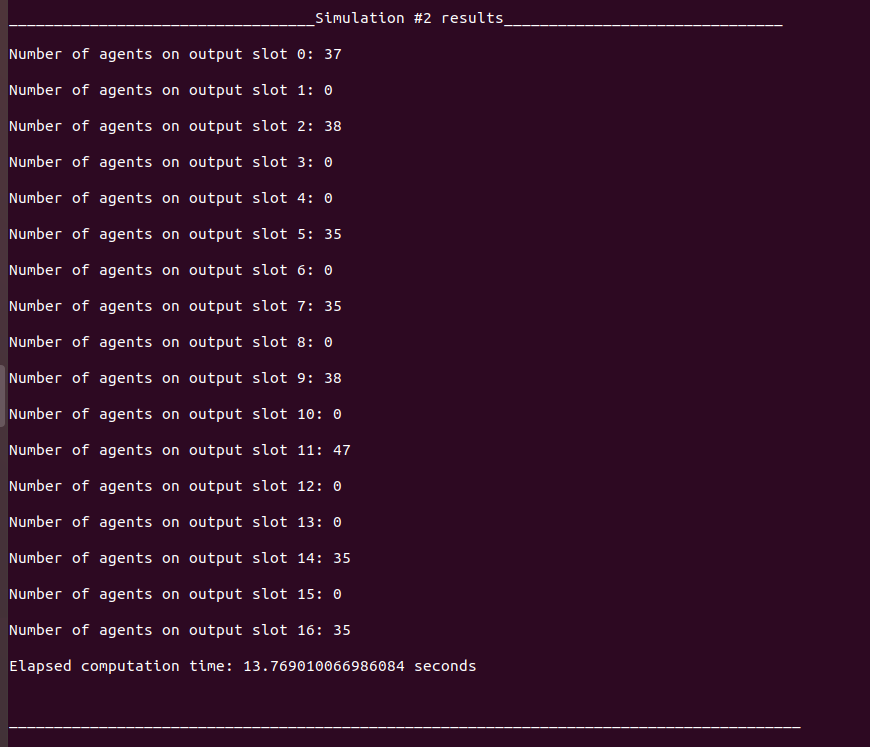
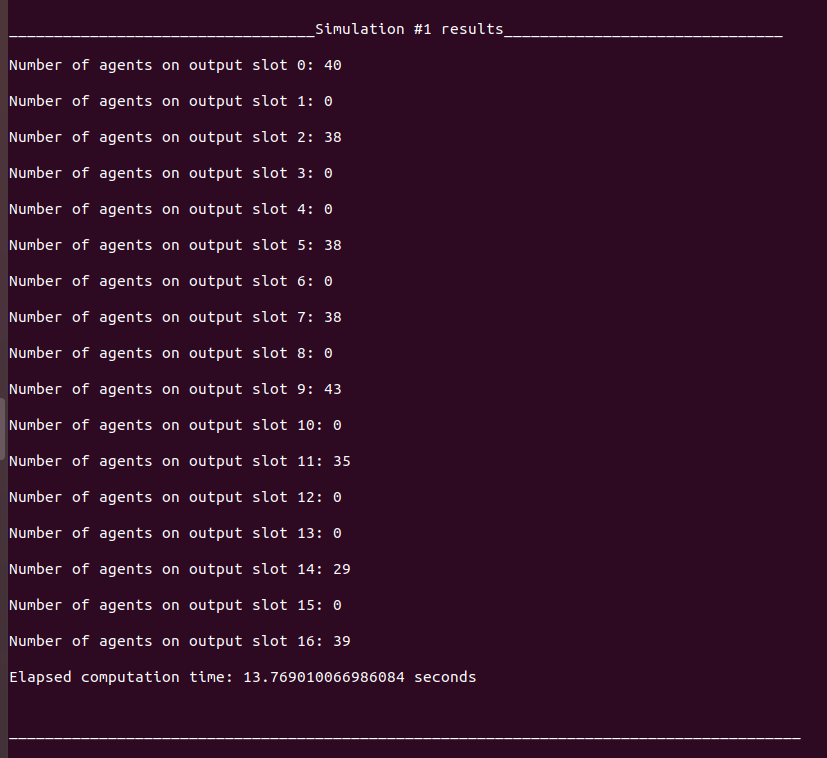
Manual SSP : manually enter the input numbers of the SSP problem

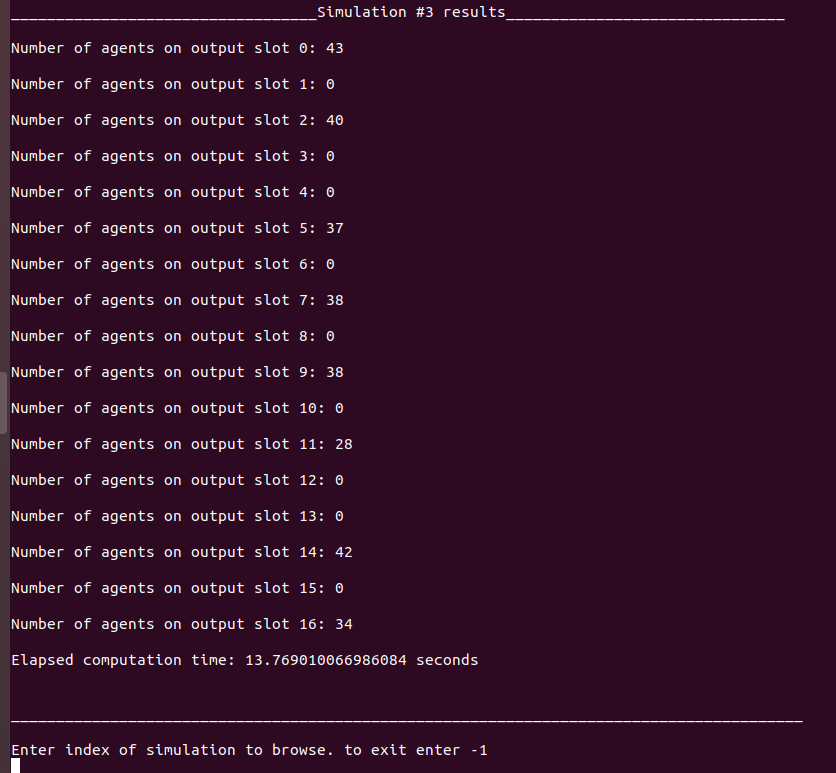
**Manual SSP:**

This mode allows the user to simulate a network that solves the SSP.

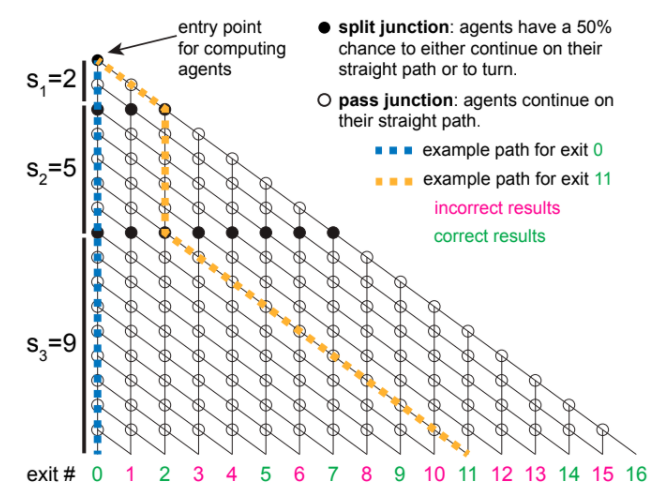
We used BCS to synthesize the computational network and simulate the runs of agents through it.







In this example we simulated the following network:



We chose a set of length 3 which contains the numbers {2, 5, 9}.

The amount of simulated agents was 300, and we chose to tag them.

In this case we used one thread for the calculation, and performed three different simulations (with the same parameters).

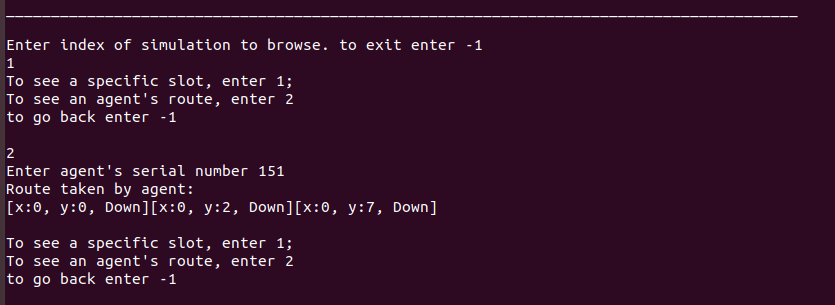
When the simulation is relatively big, increasing the number of threads may improve the run-time of the simulation. As written in Mboemo's BCS manual:

"Simulations can be run independently on separate threads, so multithreading can speed up runtimes considerably. We recommend using as many threads as you have available if the simulation is large."

Source: <https://beacon-calculus-simulator.readthedocs.io/en/latest/simulation.html>

After the simulation has ended, we can browse through the results, see how many agents ended up in each output slot, and watch each agent's full route in the network.

For example, to watch the route that agent number 151 in simulation #1 took:

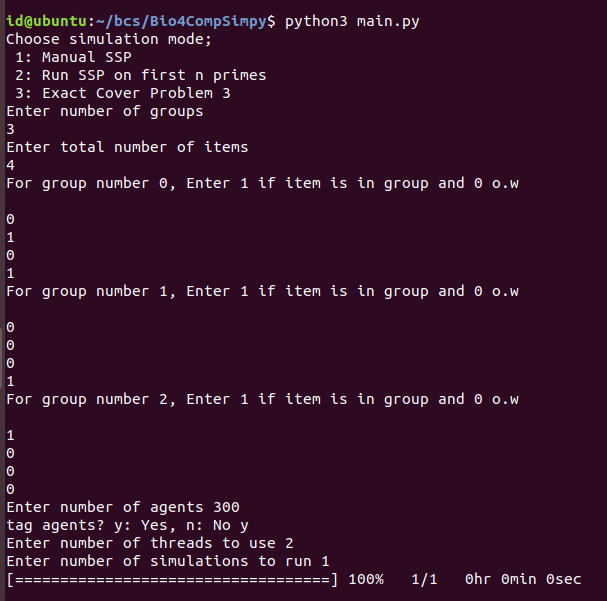


In this case we see that the agent took only down turns, meaning he chose the empty set {} so he ended up in slot 0.

**SSP on first N primes:**

This is pretty straight-forward, instead of manually choosing the numbers for the SSP, the user inputs on how many first prime numbers he wants to run the simulation on.

**Exact Cover:**

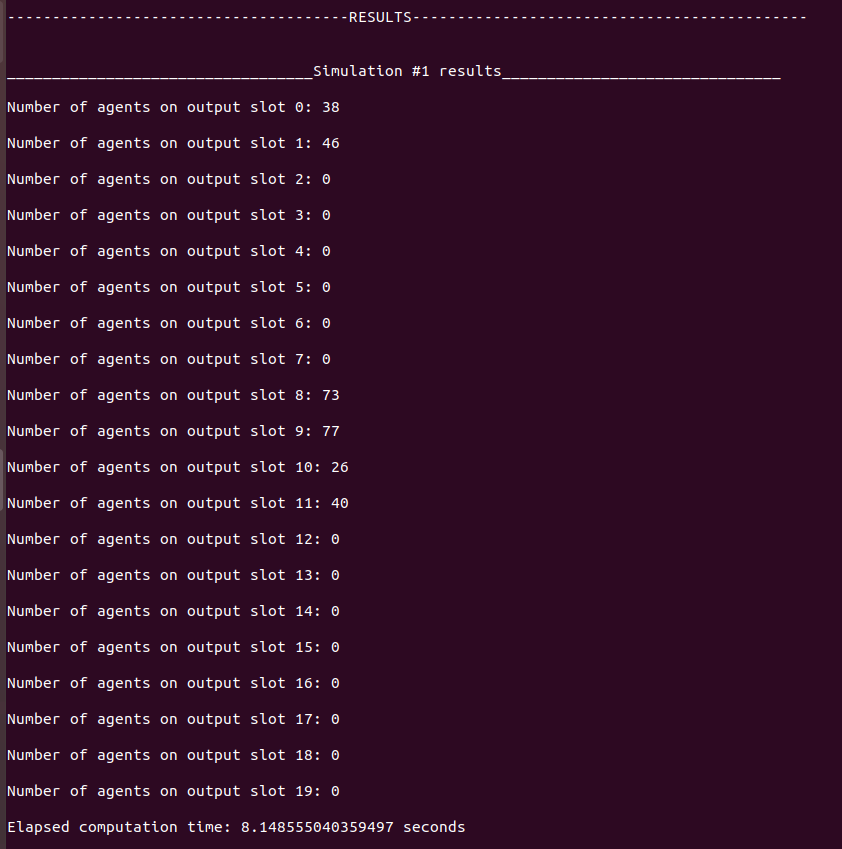


In this example we simulated a network solving the ECP with the following sets:

[1010] (=10), [1000](=8), [0001](=8) (The first digit in each group input is the LSB)

These are 3 groups with 4 possible items. Choosing the same item twice is illegal, so the network blocks some of the junctions thereby forcing agents in these junctions to take the down split, which means not choosing that specific group.

The simulation included 300 agents with tags, and was run using 2 threads.



As in the other simulation options we can browse through the results and the agents routes.

**Plotting the results**

For each simulation, the program creates a .png file of the results graph (i.e number of agents in each slot)

**Programmers manual - Code Overview:**

Structure of the working directory:

