

University of science
Vietnam National University Ho Chi Minh City



Report: Searching Solver with NetLogo
Course: Introduction to Artificial Intelligent

Lecturer:

Nguyen Ngoc Thao

Le Ngoc Thanh

Ho Chi Minh, 03/04/2022

Table of contents

I/ Members:	3
II/Assignment Plan:	3
III/Self-assessment:	3
IV/Algorithms:	3
V/Requirement 1 research:	4
VI/Experiments and comments:	10
The map	10
Key map	20
Levels	20
VII/Videos for requirements:	29
VIII/Files in the project:	29
IX/References:	30

I/ Members:

Le Nguyen Nguyen Anh - 20127438

Le Anh Kiet – 20127542

Dang Hoang Thai - 20127623

Our git repository: <https://github.com/nnaaaa/SearchingSolver>

II/Assignment Plan:

Dates	Content
15/03/2022 – 22/03/2022	Research before making the project
23/03/2022 – 26/03/2022	Making main file
27/03/2022 – 28/03/2022	Complete base of GBFS, A*, UCS, BFS, DFS
29/03/2022-30/03/2022	Fixing bugs
31/03/2022-03/04/2022	Adding levels and fixing bugs, making videos
03/04/2022 – 05/04/2022	Making video about tests case and write report.

III/Self-assessment:

No.	Criteria	Progress
1	Requirement 1	100%
2	Requirement 2: Level 1	100%
3	Requirement 2: Level 2	100%
4	Requirement 2: Level 3	100%
5	Generate at least 5 test cases for each level with different attributes. Describe them in the experiment section of your report. Videos to demonstrate.	100%
6	Report your algorithms, experiments with some reflection or comments.	100%

IV/Algorithms:

+GBFS: Take the node that has the smallest h value in the frontier and generate the child of that node, we'll check if the child node is the destination node then we stop, if not, we push it in the frontier and run until the frontier is empty.

+A*: take the node that has the smallest path cost + h value from the frontier, if it is the destination then we stop, if not, we generate the childs of it, if the child is in the frontier

and has $g+h$ smaller than it in the frontier then we replace it, we will run until the frontier is empty. (path cost is g and h is heuristic value)

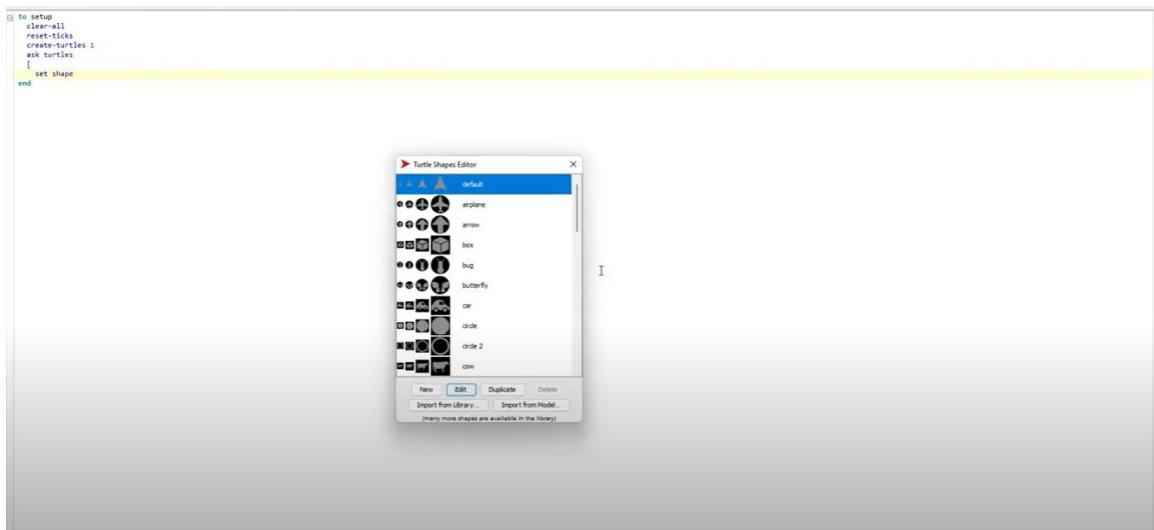
+UCS: take the node that has the smallest g value from the frontier, if it is the destination then we stop, if not, we generate the childs of it, if the child is in the frontier and has g smaller than it in the frontier then we replace it, we will run until the frontier is empty. (path cost is g and h is heuristic value)

+BFS: Take the first node in the frontier and generate the child of that node, we'll check if the child node is the destination node then we stop, if not, we push it in the frontier and run until the frontier is empty.

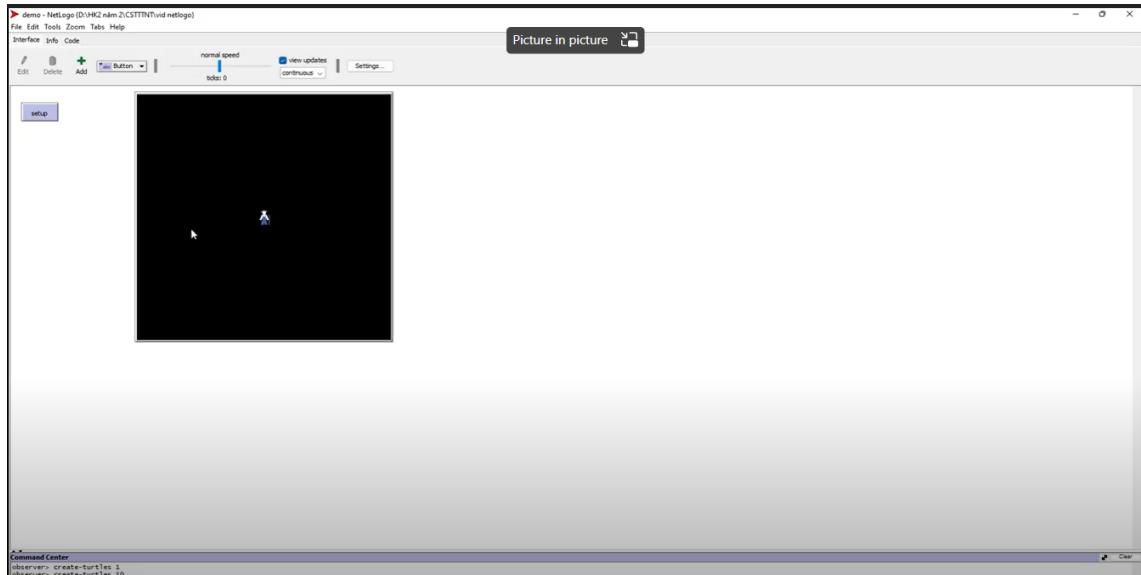
+DFS: Take the first node in the frontier and generate the child of that node, we'll check if the child node is the destination node then we stop, if not, we put it in the top of the frontier and run until the frontier is empty.

V/Requirement 1 research:

_Show agent shapes:



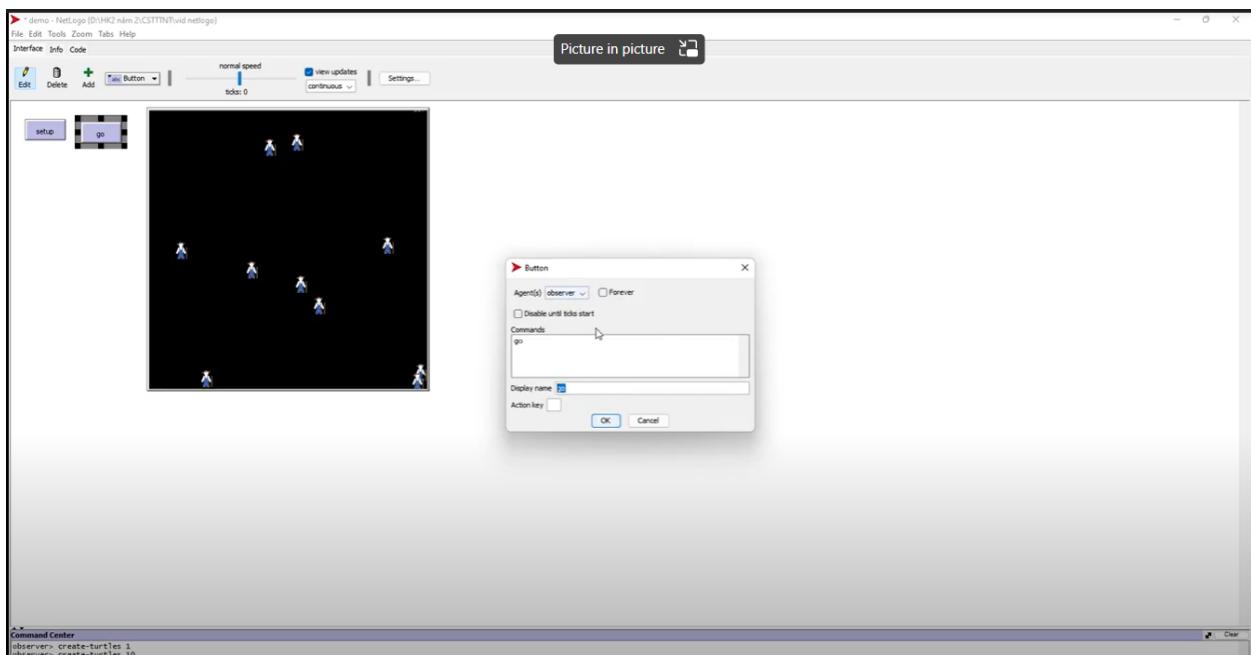
Picking a shape for the agents



After picked a shape (default is a arrow)

We will use the “set shape” command to set it’s icon, we will use this to set the agent as a humand and a house symbolize for agents and destinations.

_Show buttons:



Creating a button with the go function

The button is where user can click on it to trigger a function, we will use it for several purpose.

_ Showing running parameter of agents:

```
turtles-own [pathogens?]
to setup
  clear-all
  reset-ticks
  create-turtles num-of-farmer
  ask turtles
  [
    set shape "person farmer"
    set size size-of-farmer
    set pathogens? false
    switch-color
    setxy random-xcor random-ycor
  ]
end

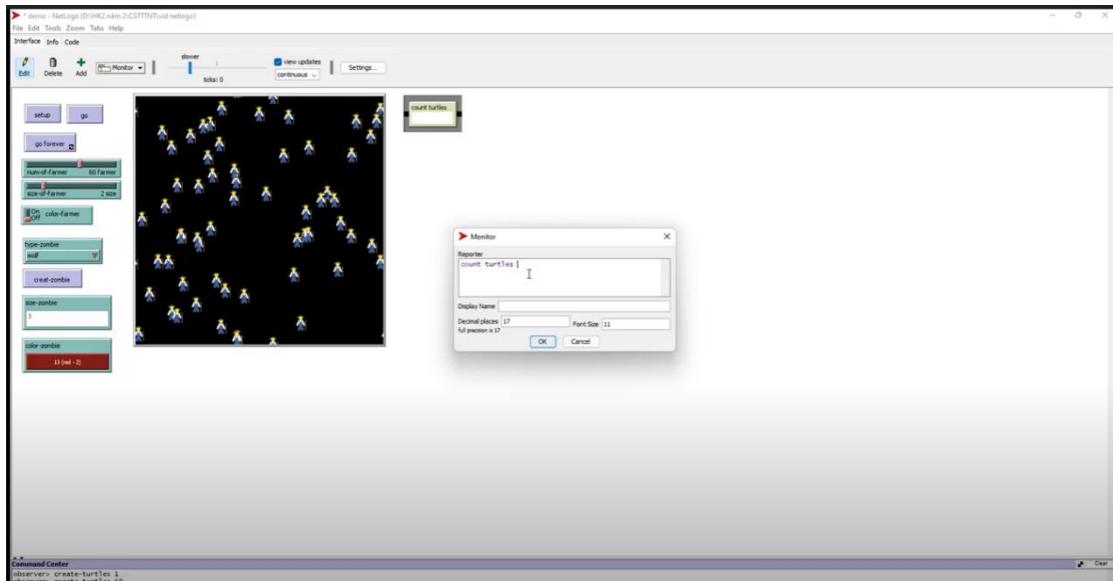
to switch-color
  ifelse color-farmer
  [
    set color white
  ]
  [
    set color yellow
  ]
end

to go
  ask turtles
  [
    ifelse coin?
    [
      right random 60] [ left random 60]
      forward random 4
      if any? turtles-here with[pathogens?]
      [set pathogens? true]
      if pathogens?
      [
        set size size-zombie
        set color color-zombie
      ]
    ]
  ]
end

to-report coin?
  report random 2 = 0
end

to create
  ask one-of turtles
  [
    set size size-zombie
    set color color-zombie
    set pathogens? true
    if type-zombie = "person farmer"
    [set shape "person farmer"]
    if type-zombie = "wolf"
    [set shape "wolf"]
    if type-zombie = "sheep"
    [set shape "sheep"]
  ]
end
```

Example code of zombie infection

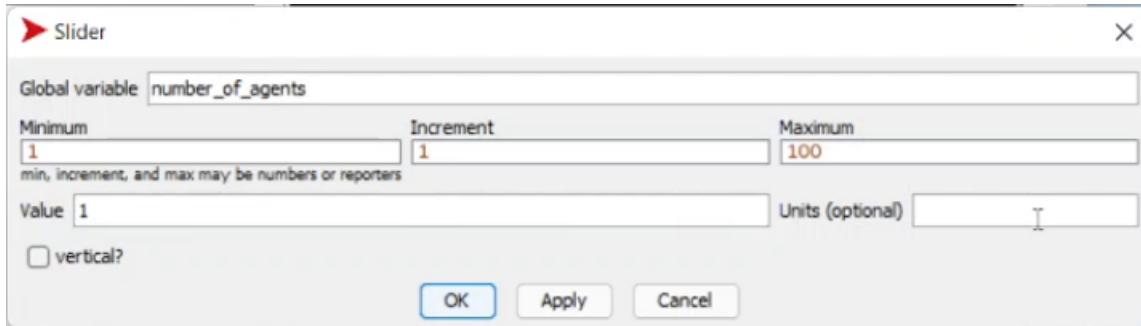


Creating the plot for counting zombie

The plot will count the size of variable, we will use this for determine which algorithm will run fastest.

_ Create utilities:

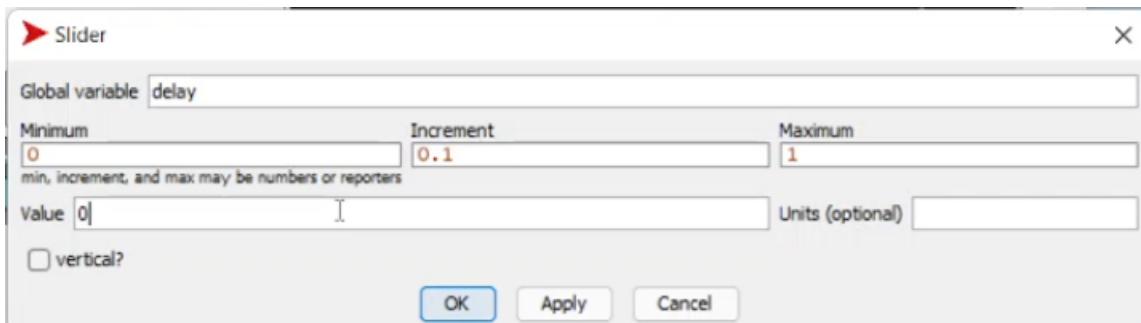
+Slider:



Window when create a slider for setting the number of agents

```
breed [commuters commuter]
to generate-commuters
  ask commuters[die]
  create-commuters number_of_agents[
    set shape "turtle"
    setxy random-xcor random-ycor
  ]
end
```

Code for using the variable number_of_agents



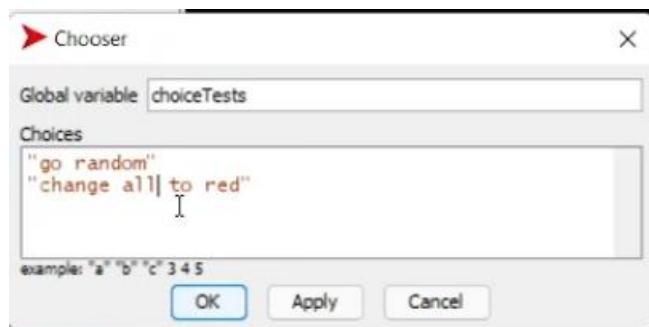
Parameter for using slider as a delay indicator

```
to go
  wait delay
  ask commuters[
    forward random 5
  ]
end
```

Code for use the variable delay

A slider is a bar that create a global variable and change it according to where on the bar that the user drag, in this project, we will use it for change number of agent and speed of the program.

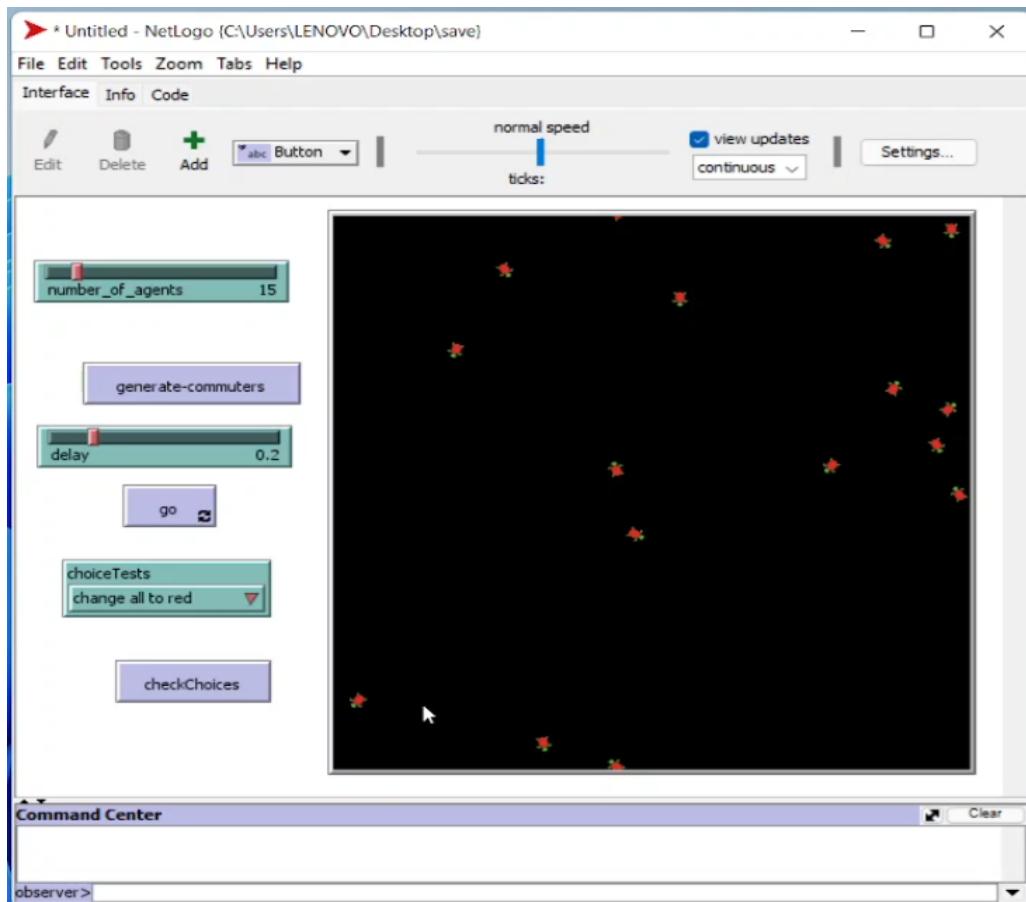
+Chooser



Window for create a example of the chooser

```
to checkChoices
  ask commuters[
    if choiceTests = "go random" [forward random 5]
    if choiceTests = "change all to red" [set color red]
  ]
end
```

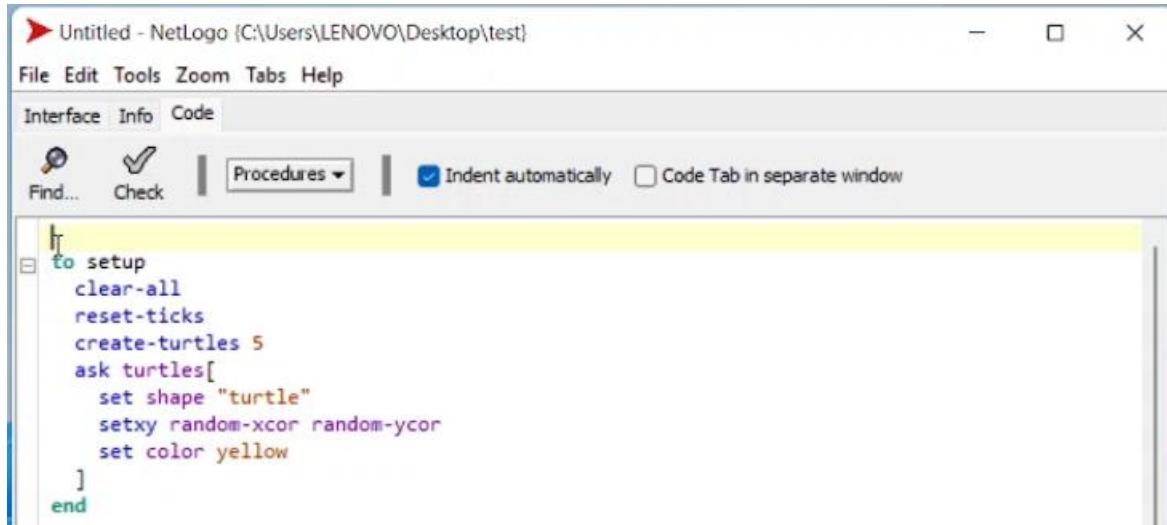
Code for using the variable choiceTests



Result of the example

A chooser is a dropdown list that the user can pick an option, we will use this for change the map and change the algorithms when the agent start to walk on the returned path.

_Color agent and the environment:



The screenshot shows the NetLogo interface with the 'Code' tab selected. The code in the editor is:

```
to setup
  clear-all
  reset-ticks
  create-turtles 5
  ask turtles[
    set shape "turtle"
    setxy random-xcor random-ycor
    set color yellow
  ]
end
```

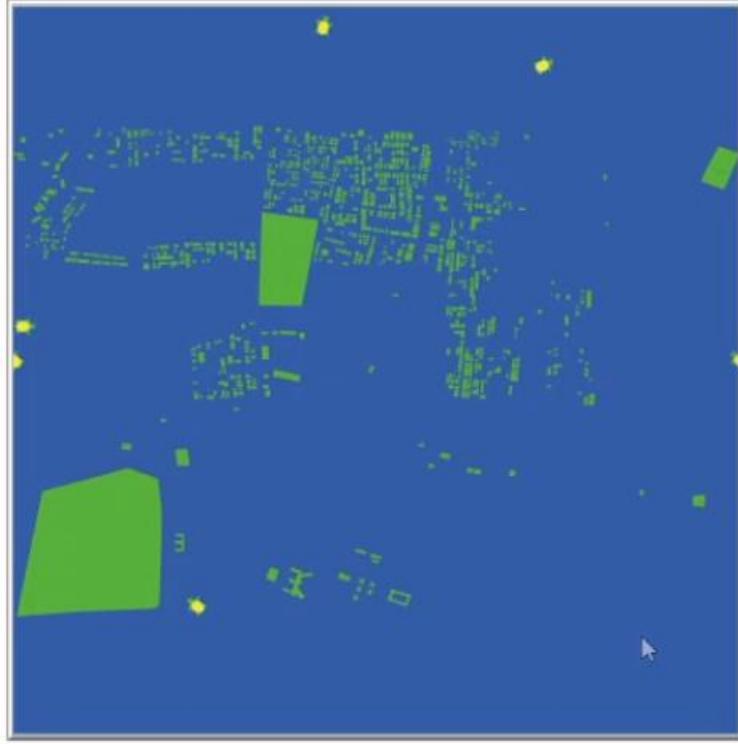
Example code for change color of all agent to yellow

```
| to setup
|   clear-all
|   reset-ticks
|   create-turtles 5
|   ask patches [set pcolor blue]
```

Example code for change color of the background

```
| to createBuildings
|   set buildings gis:load-dataset "4b.shp"
|   gis:set-drawing-color green gis:fill buildings 1
|   gis:set-world-envelope gis:envelope-of buildings
| end
```

Example for import map and change color of the building



Result of example

We will use the default command of netLogo and command from the extension gis (because we will use the open street map and qgis in this project) in order to mark the algorithms and the buildings.

VI/Experiments and comments:

Note: With level 1 and 2 we will first color the road it went through to find the route and when the path is return it will change color, when the agent walk it will delete the parts of return part respectively. Level 3 we will only change the color of the agent and draw the return path. Comments on every test cases will be given through the videos.

The map

_Maps we used in the project contain 5 maps:

+Tokyo - Japan:



Area in Tokyo we took from far above

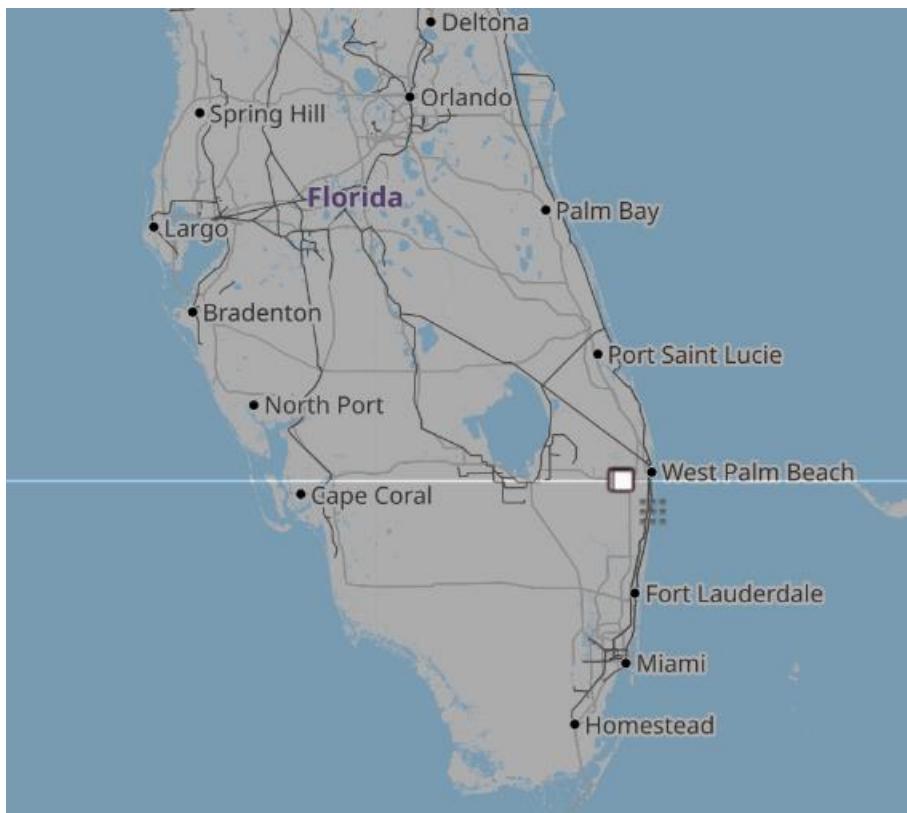


Details of the area we took in Tokyo

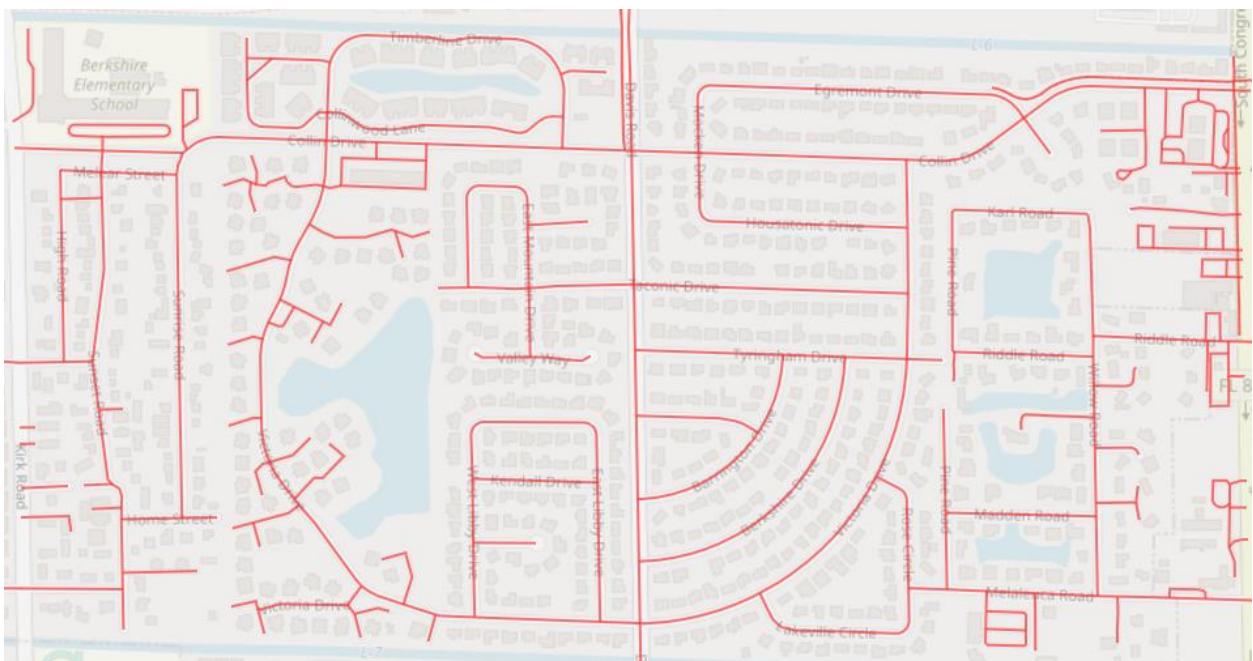


Tokyo map in NetLogo by using gis extensions

+Florida – America:



Area in Florida we took from far above

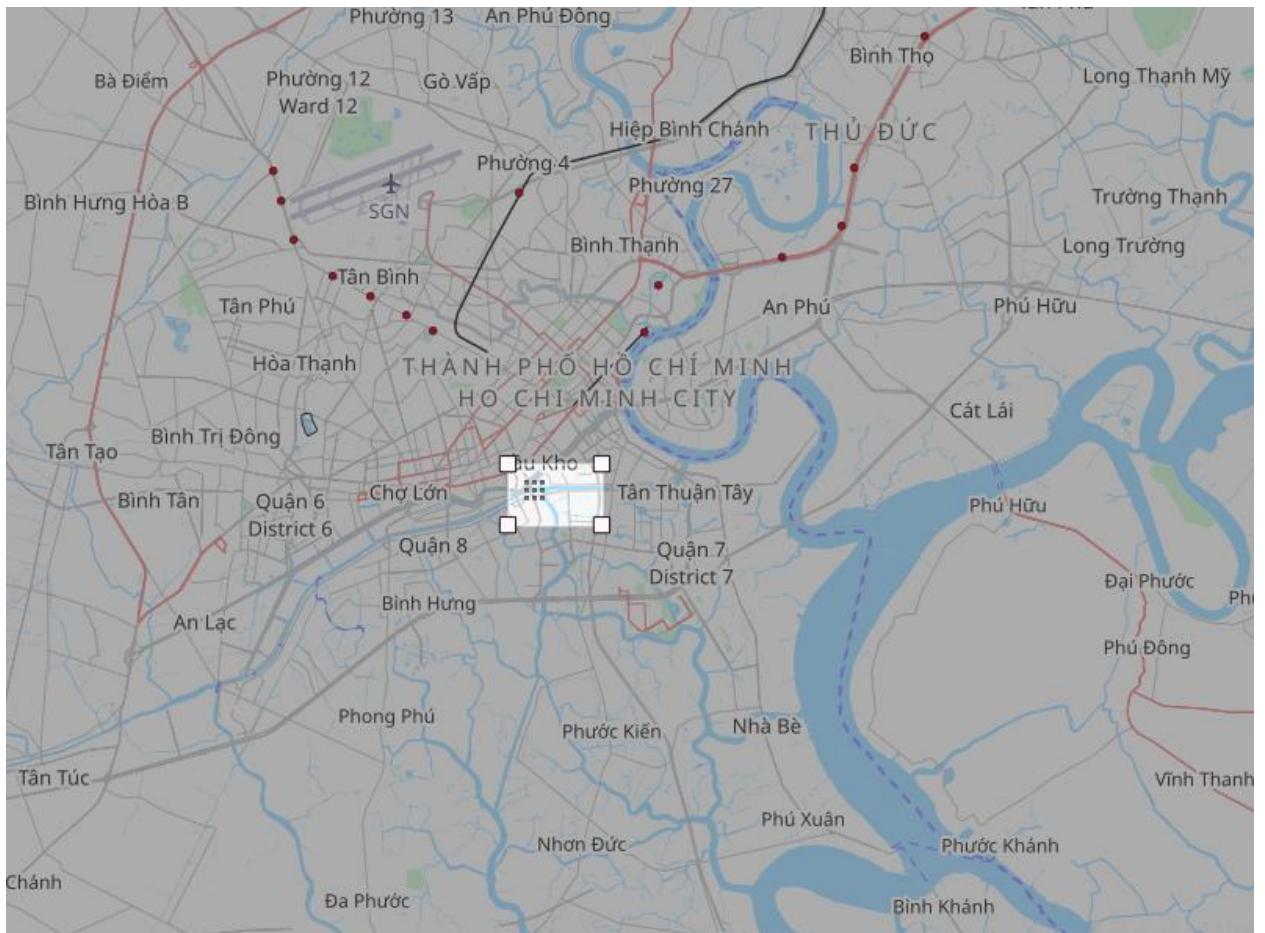


Details of the area we took in Florida



Florida map in NetLogo by using gis extensions

+HoChiMinh – Vietnam:



Area in Ho Chi Minh we took from far above

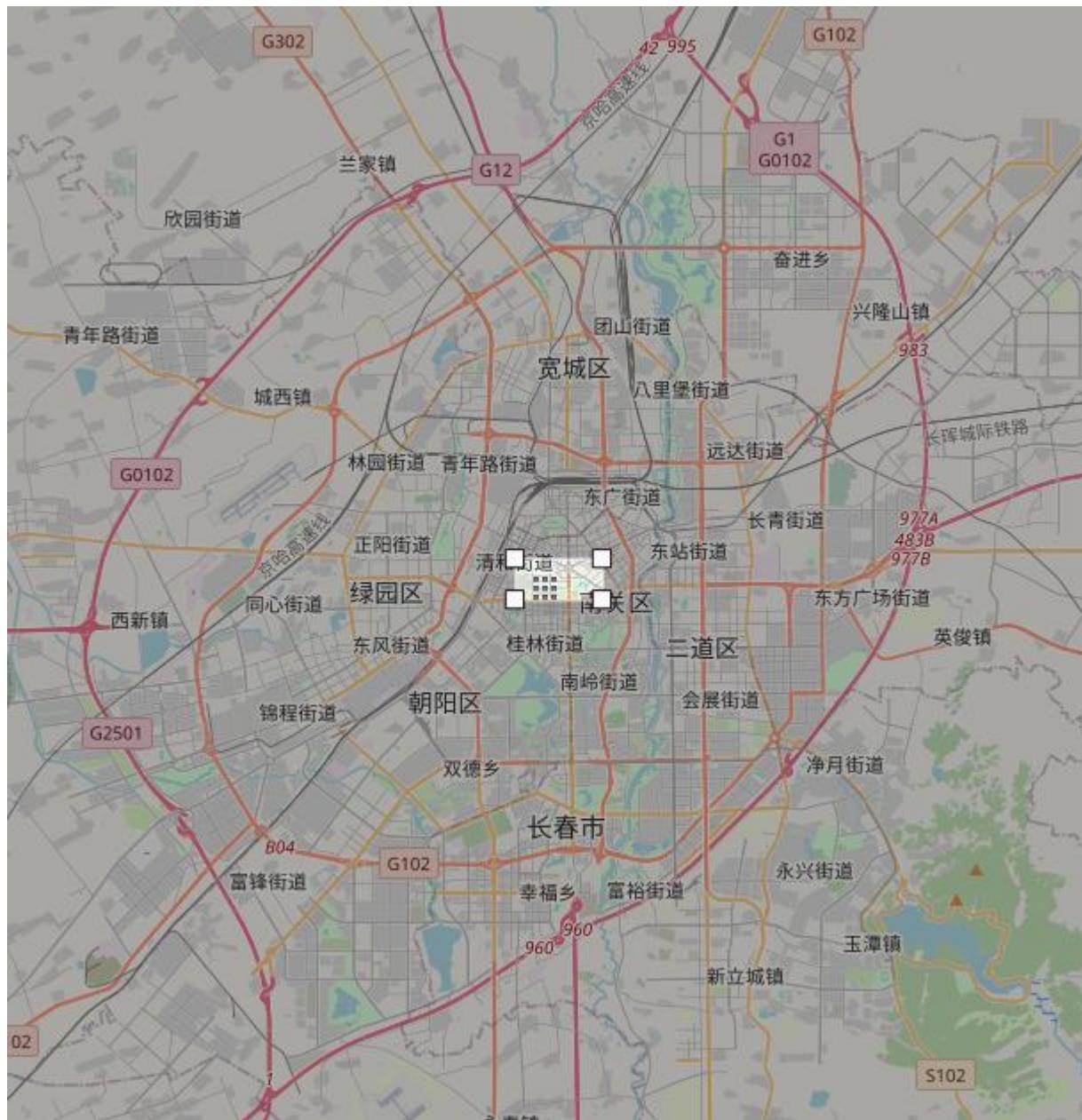


Details of the area we took in Ho Chi Minh (District 7)

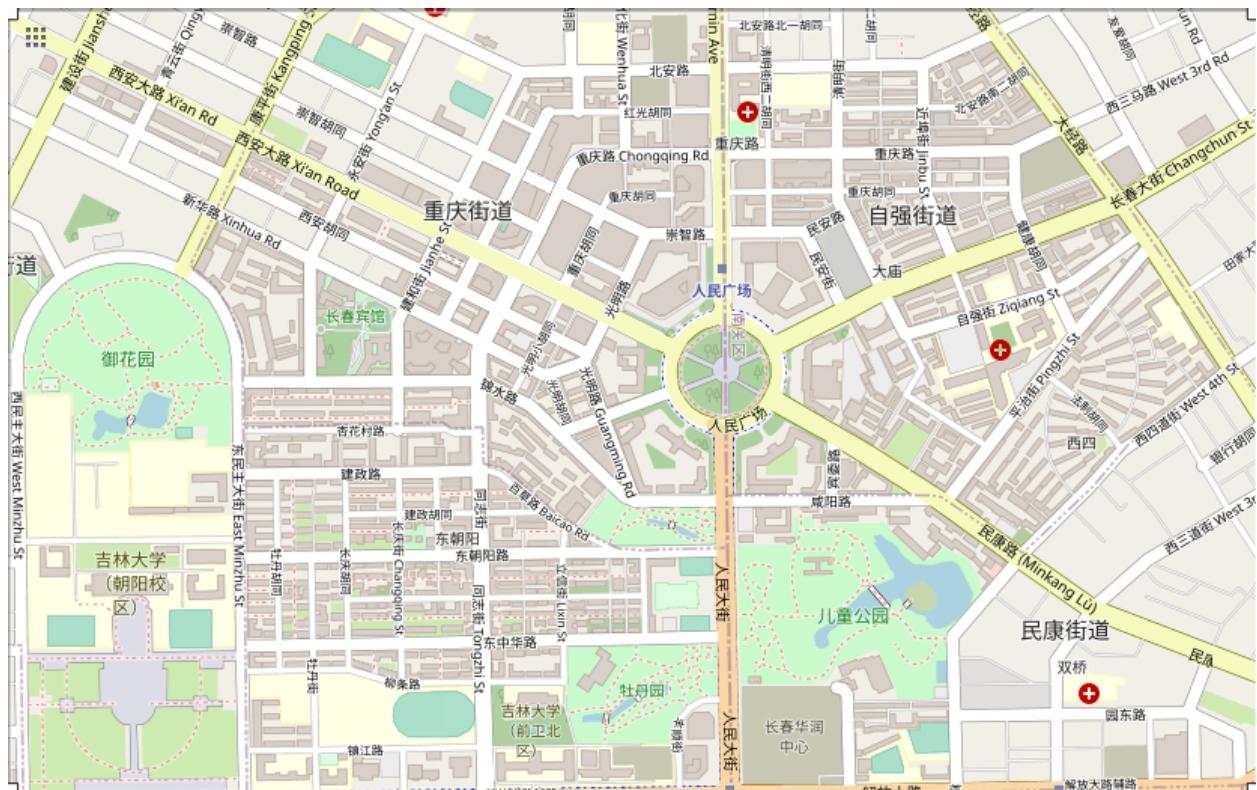


Ho Chi Minh map in NetLogo by using gis extensions

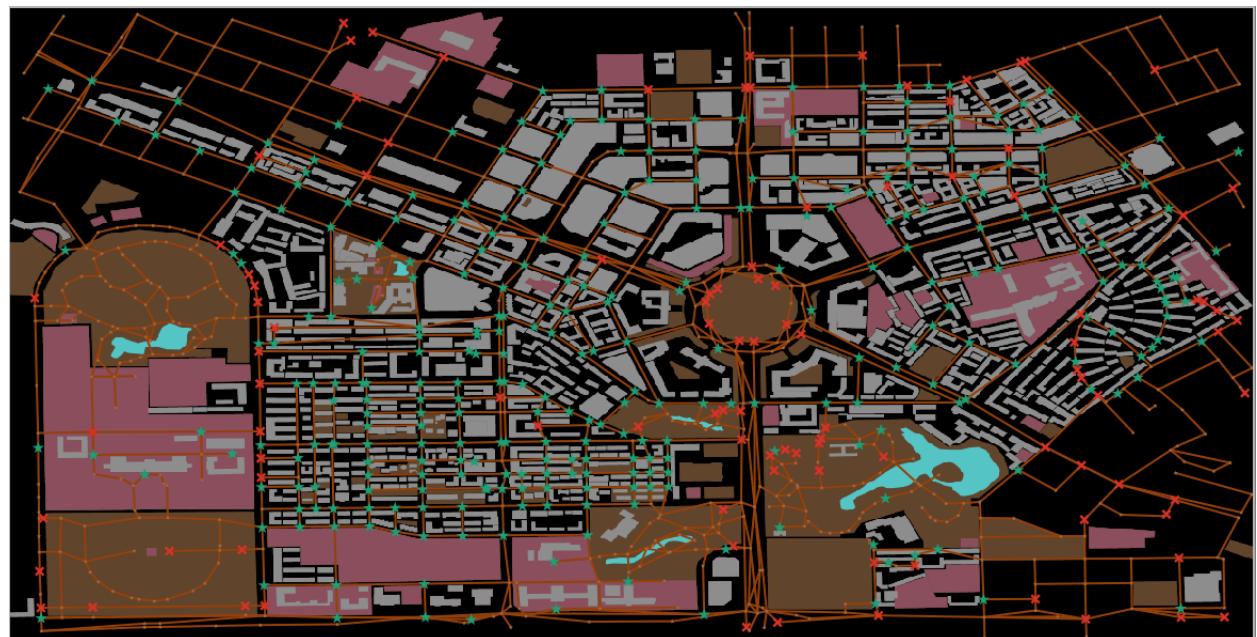
+Changchun – China:



Area in Changchun we took from far above

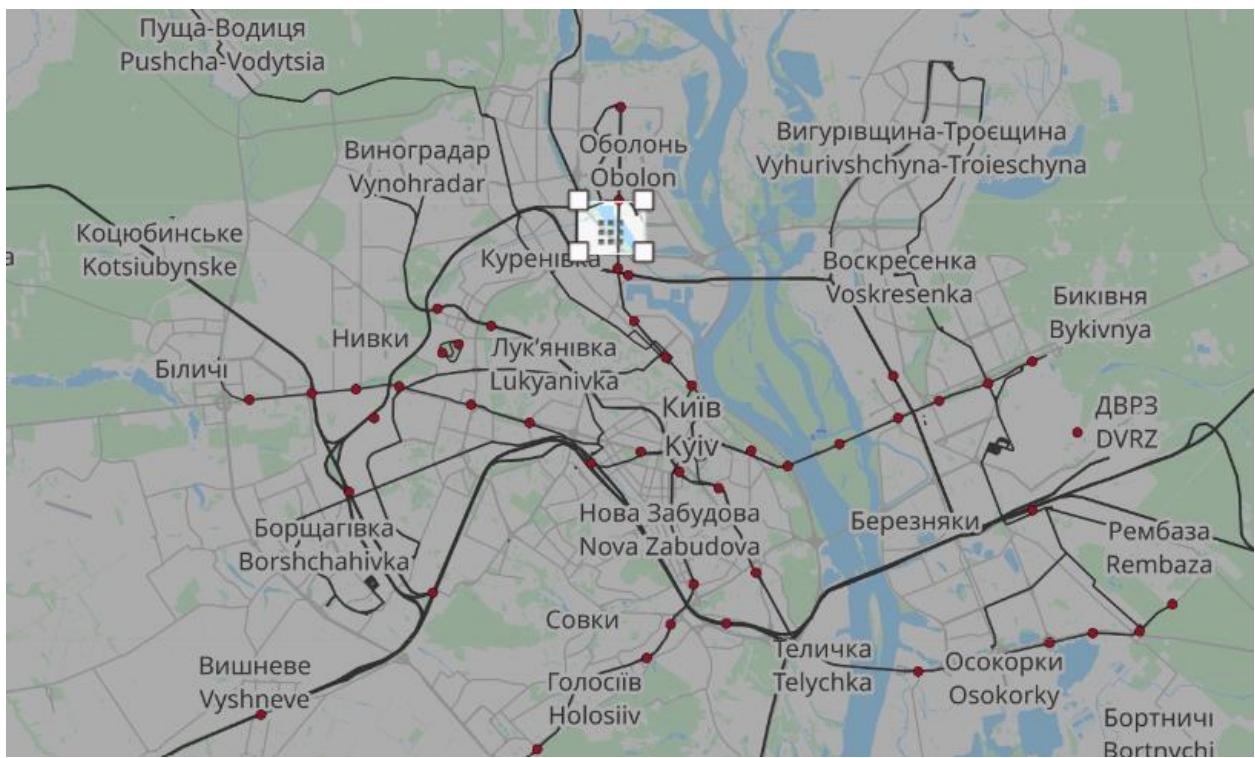


Details of the area we took in Changchun

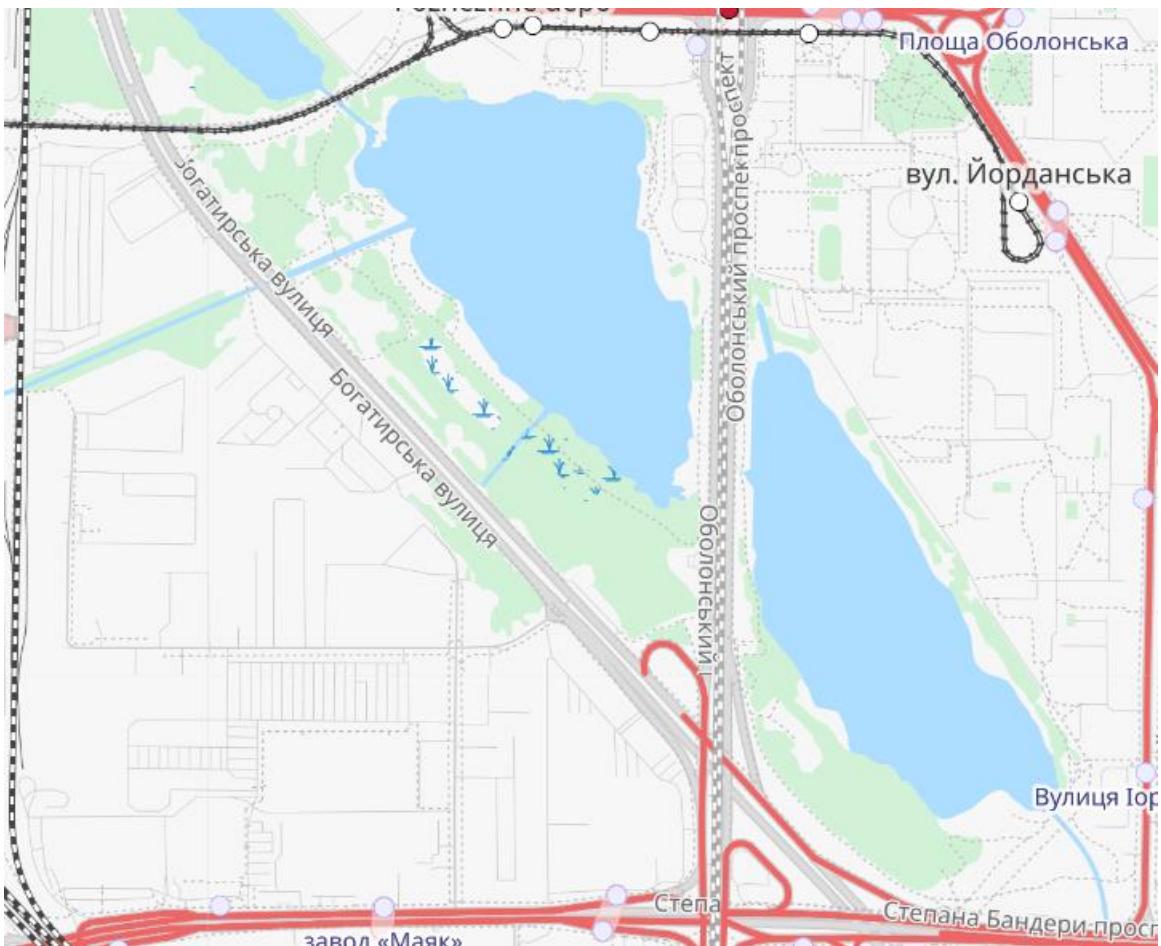


Changchun map in NetLogo by using gis extensions

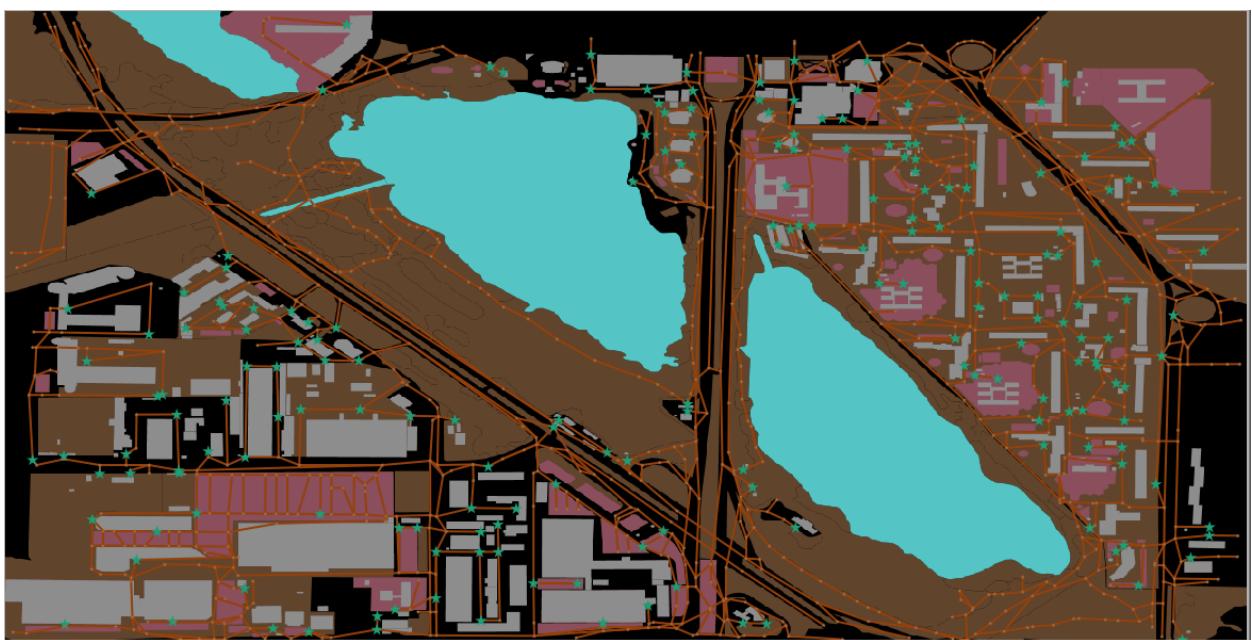
+Kyiv – Ukraine:



Area in Kyiv we took from far above

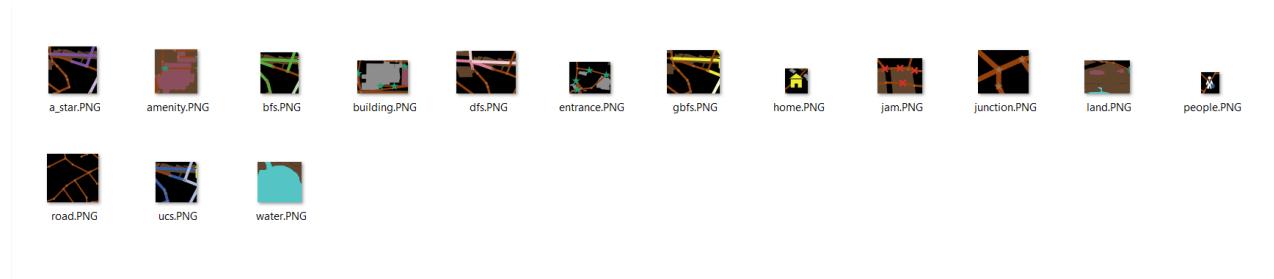


Details of the area we took in Kyiv



Kyiv map in NetLogo by using gis extensions

Key map

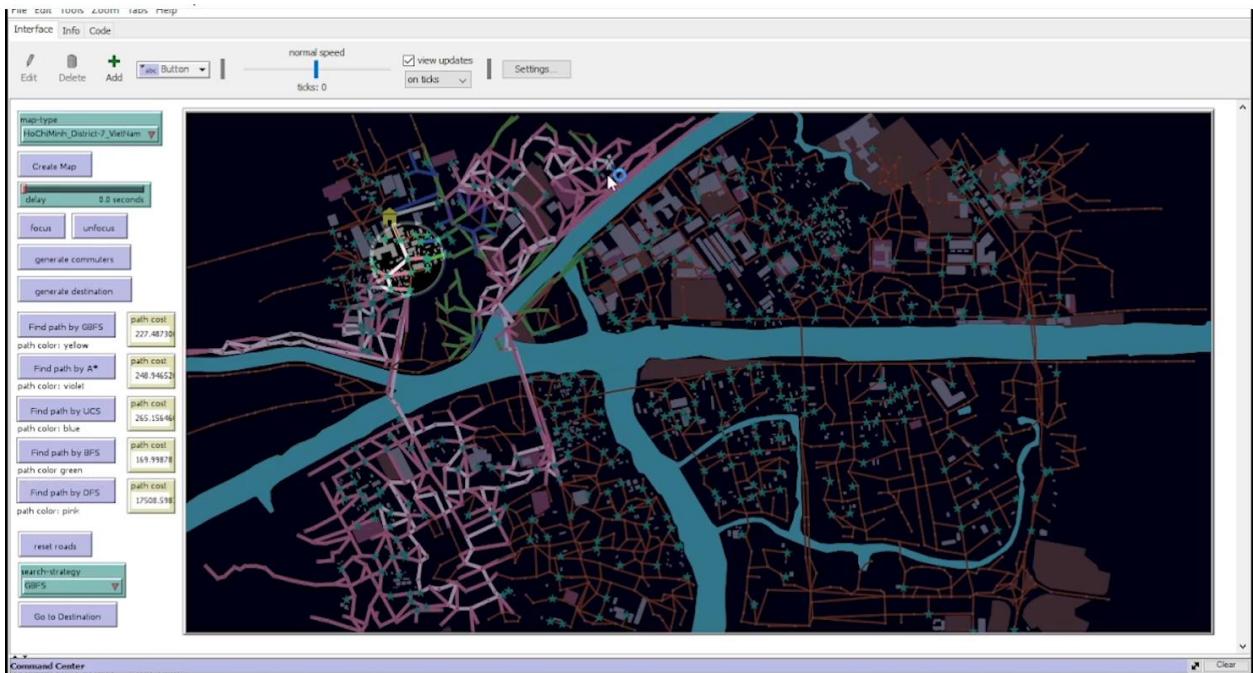


Key maps we used in the project

Levels

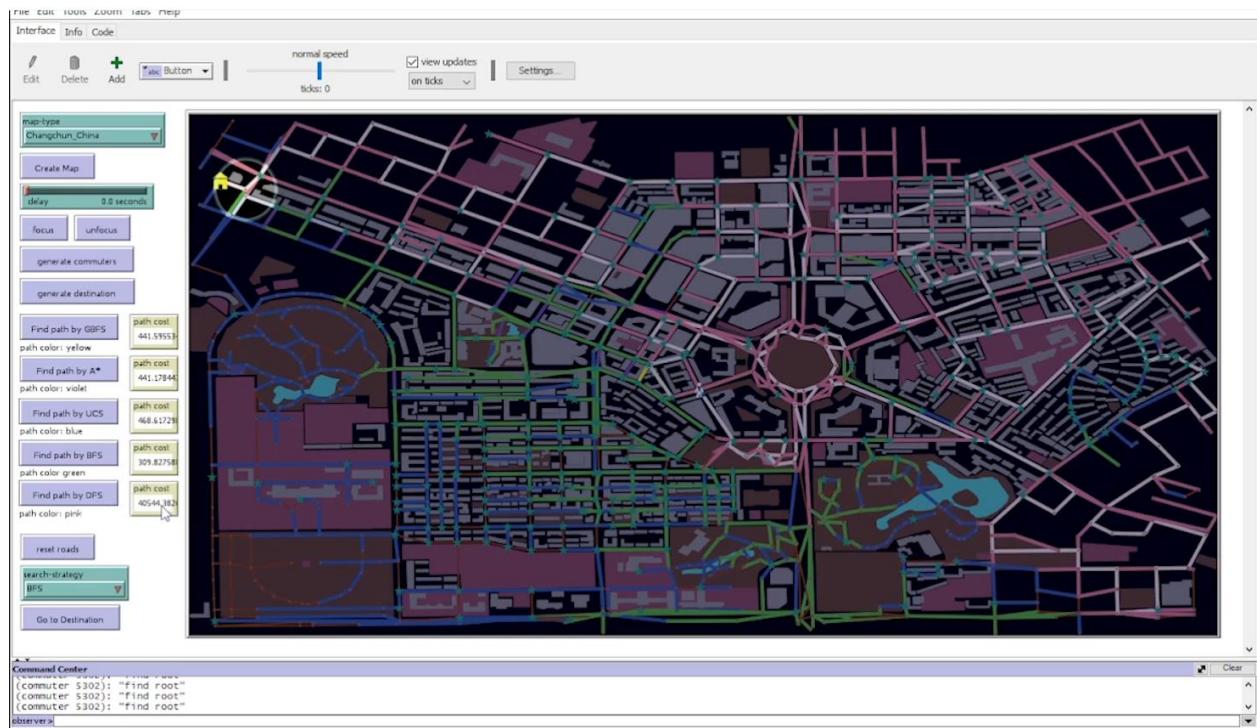
_Level 1:

+Test case 1 + 2 + 3 + 4 + 5:



Result for 5 algorithm on Ho Chi Minh map with 1 agent and 1 destination

+Extra 5 test for Changchun map:

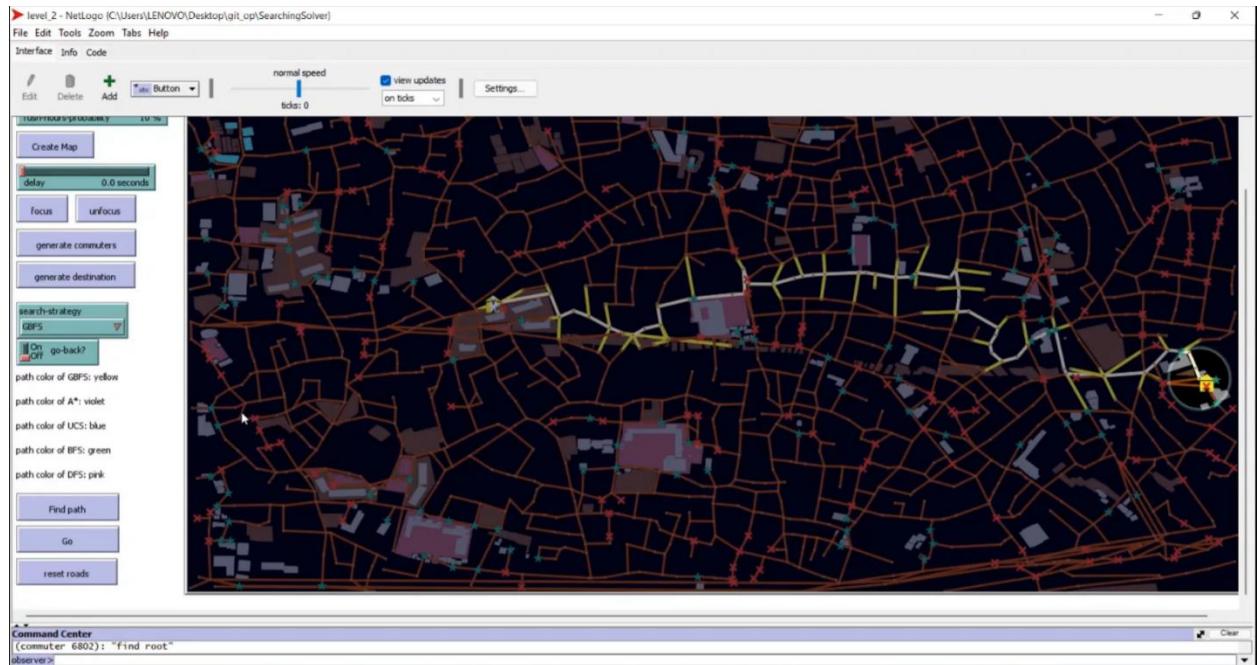


Result for 5 algorithm on Changchun map with 1 agent and 1 destination

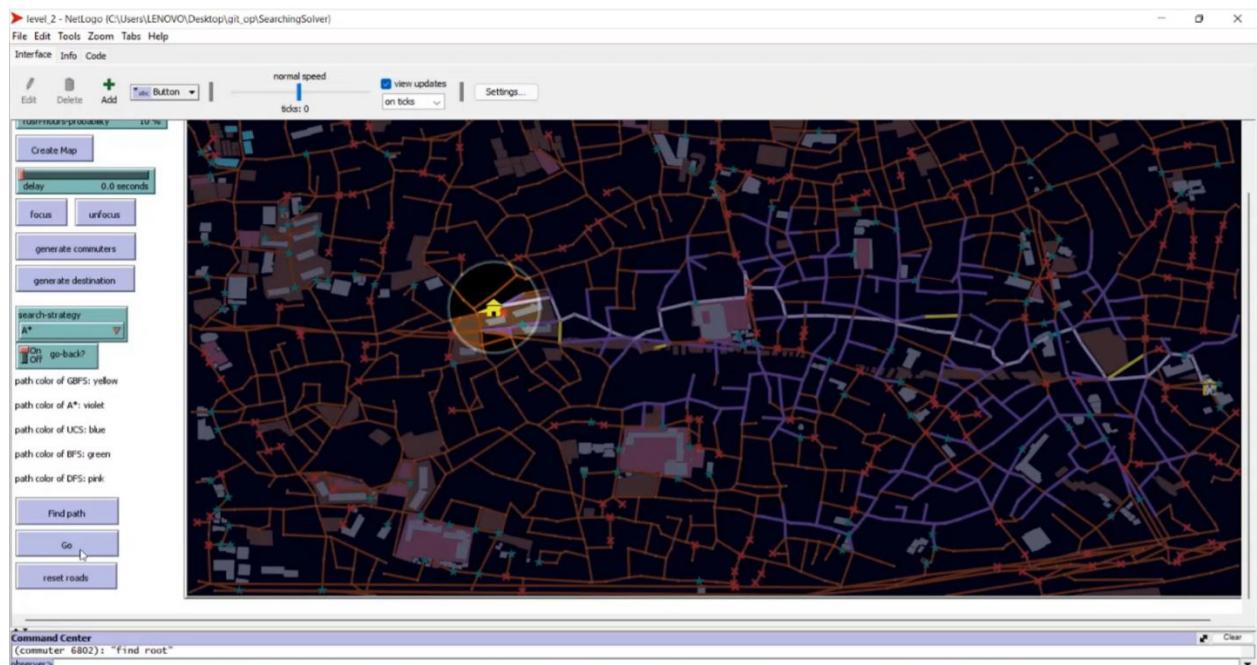
+Comments on level 1: Despite of the fact that A* and GBFS is inform search algorithms, their path cost is greater than BFS. But inform search often have path cost lower than the others. I think DFS which is one of uninform search have the longest path cost and the best time-consuming search strategy. It's not difficult to find out that in our video. It always take nearly entire vertices in map.

Level 2:

+Test case 1:

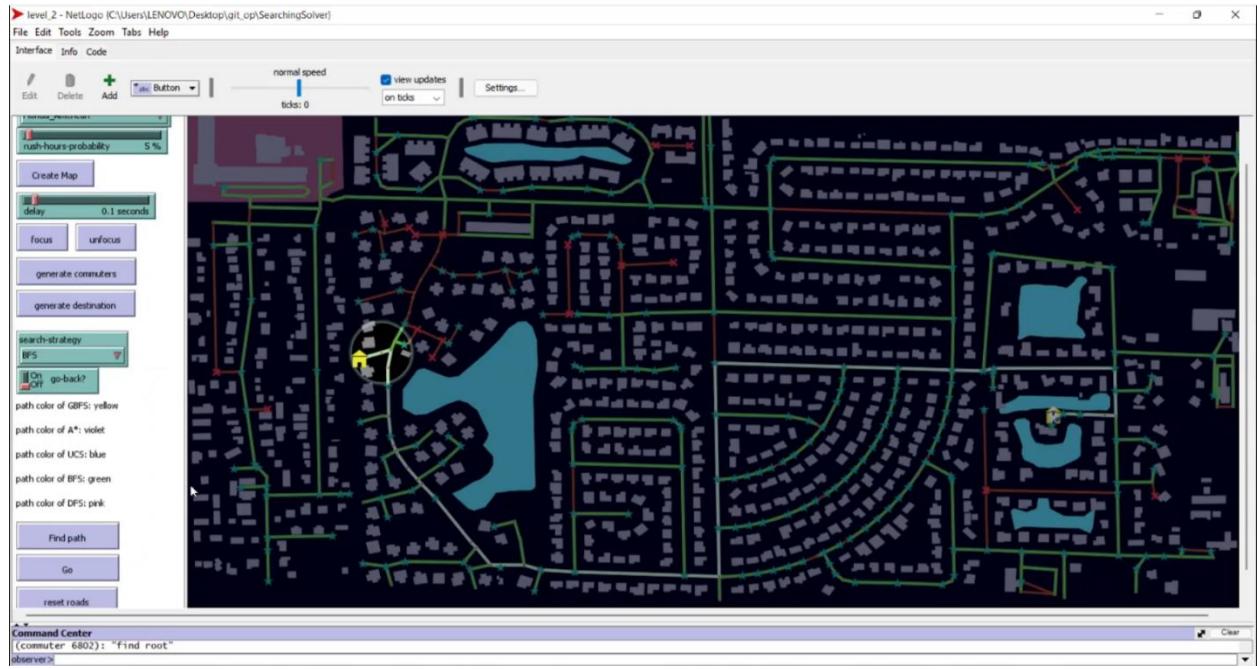


Result of returned path from start to end using GBFS on Tokyo map

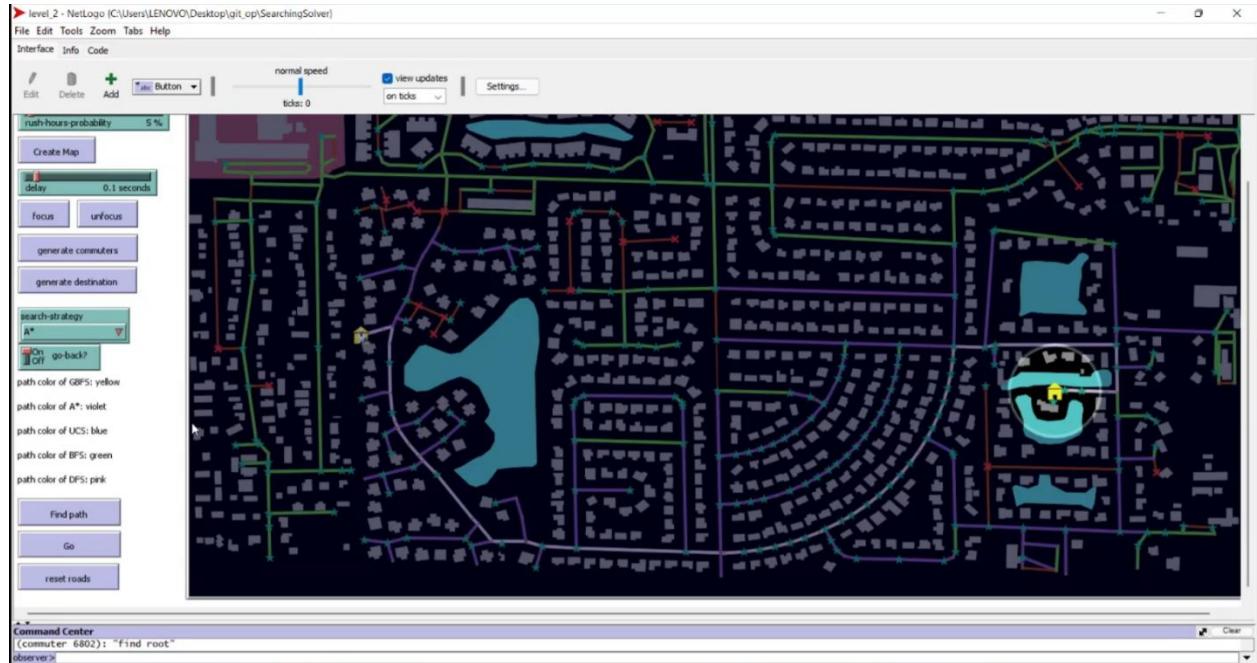


Returned path from end to start using A* on Tokyo map

+Test case 2:

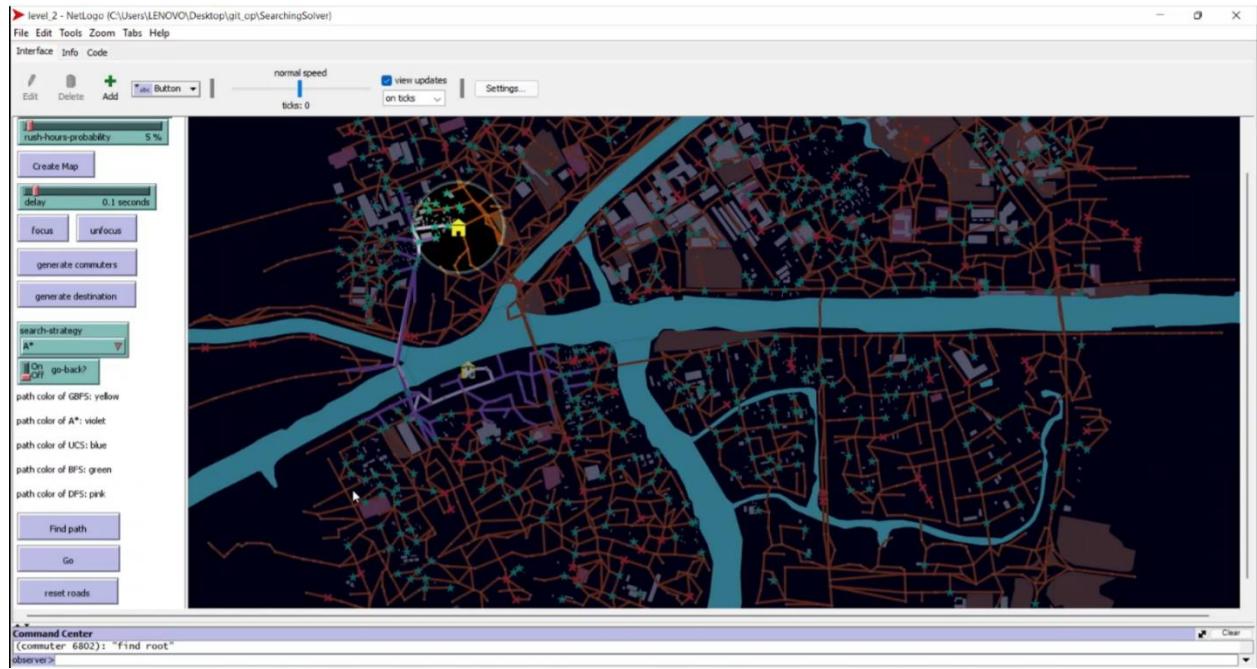


Returned path from start to end using BFS on Florida map

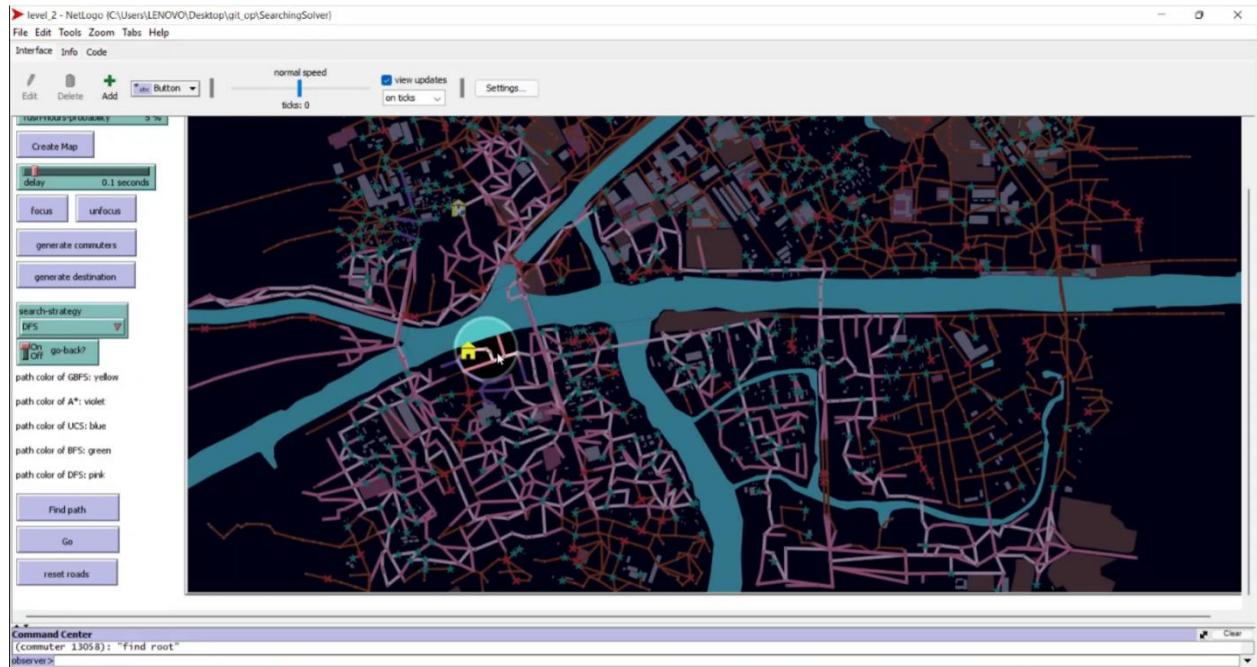


Returned path from end to start using A* on Florida map

+Test case 3:

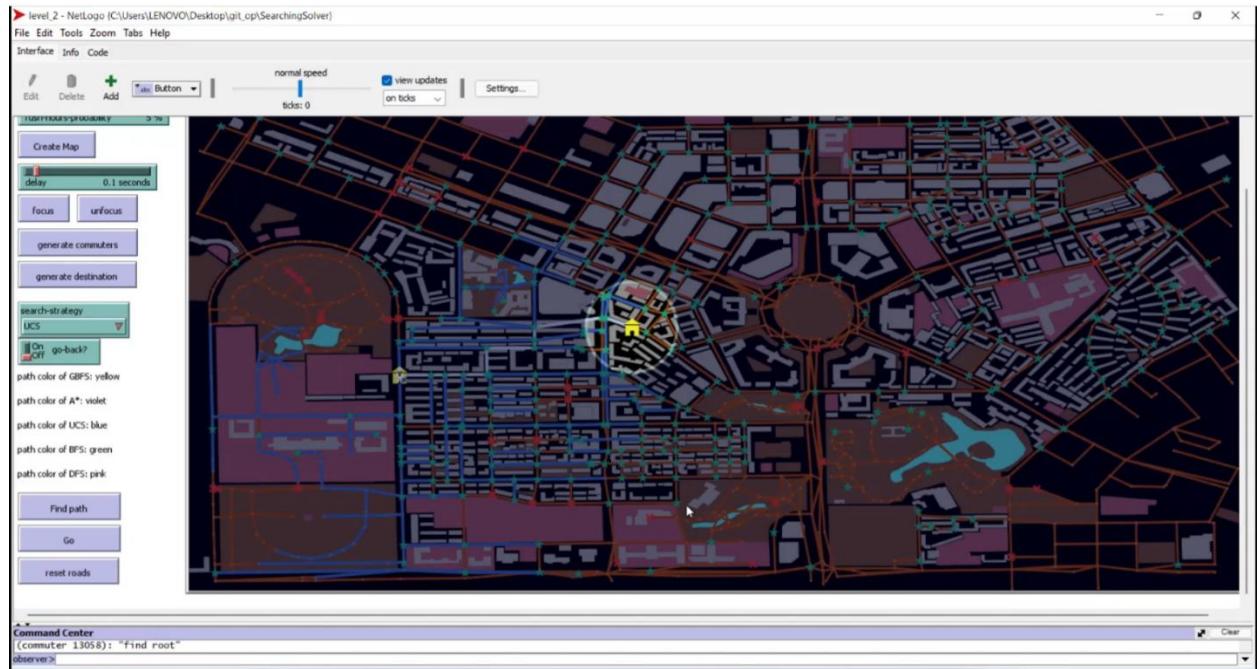


Returned path from start to end using A* on Ho Chi Minh map

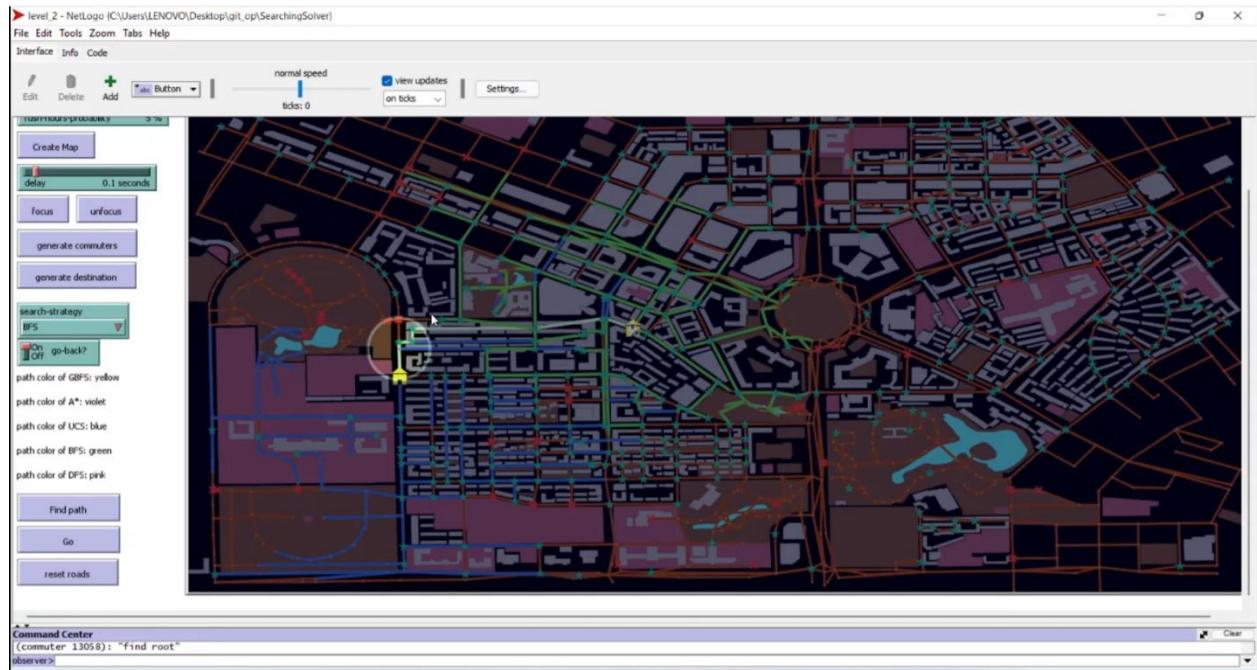


Returned path from end to start using DFS on Ho Chi Minh map

+Test case 4:

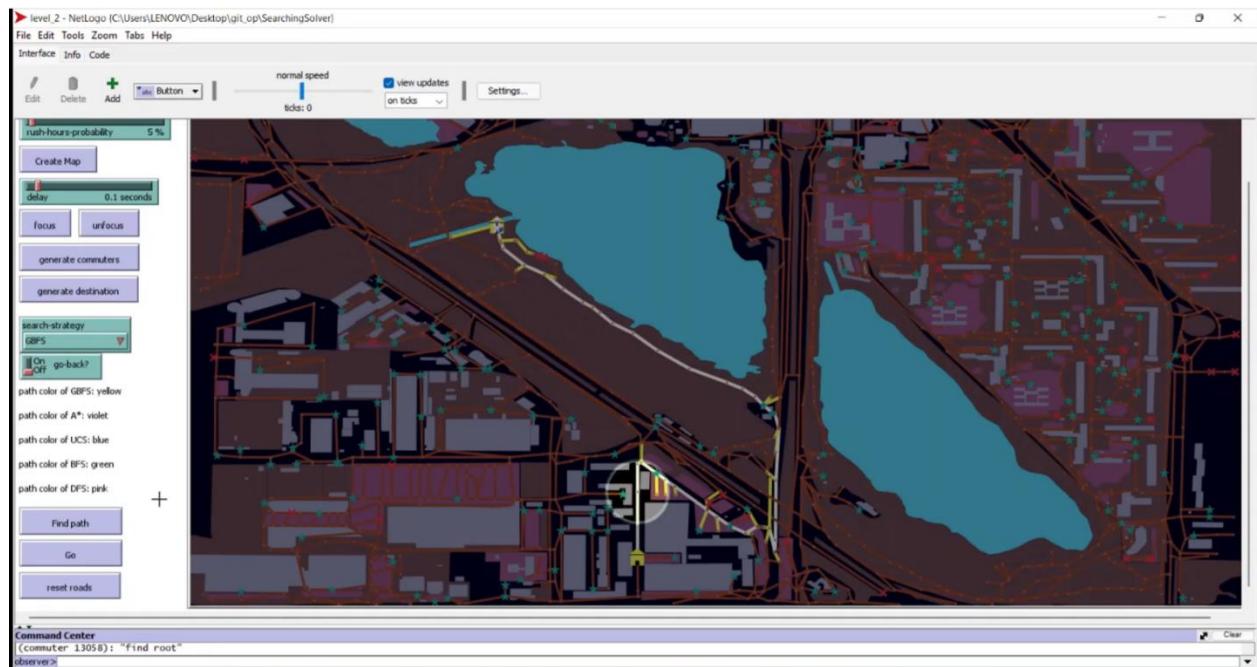


Returned path from start to end using UCS on Changchun map

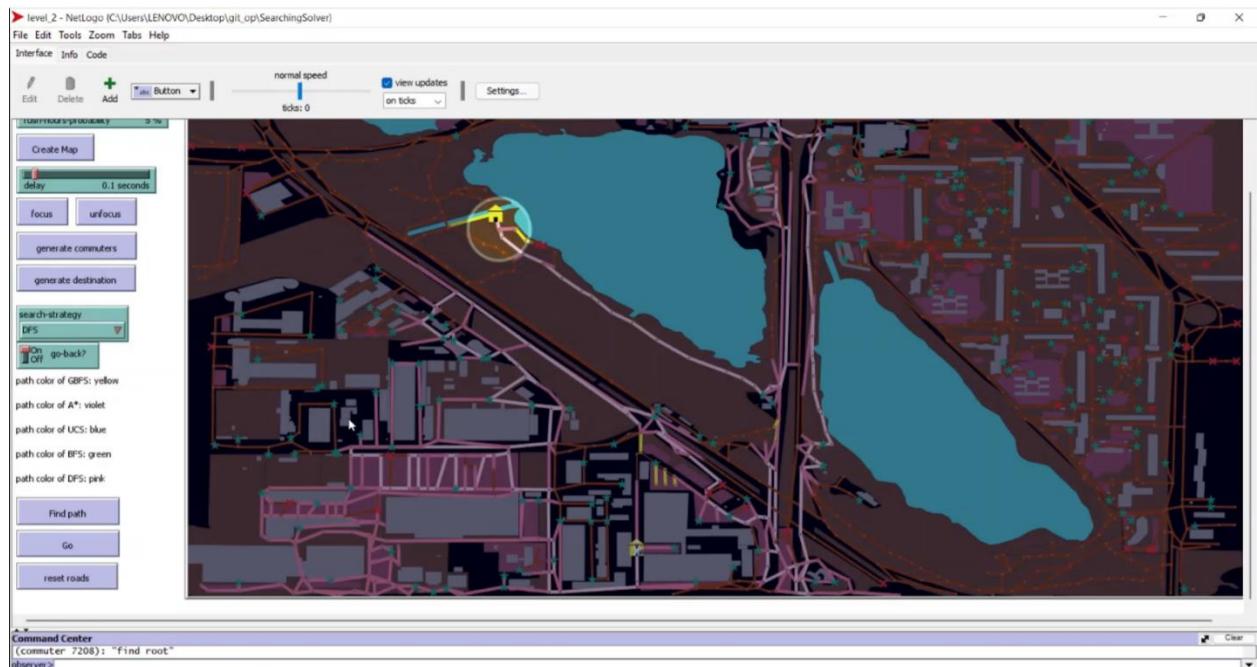


Returned path from end to start using BFS on Changchun map

+Test case 5:



Returned path from start to end using GBFS on Kyiv map



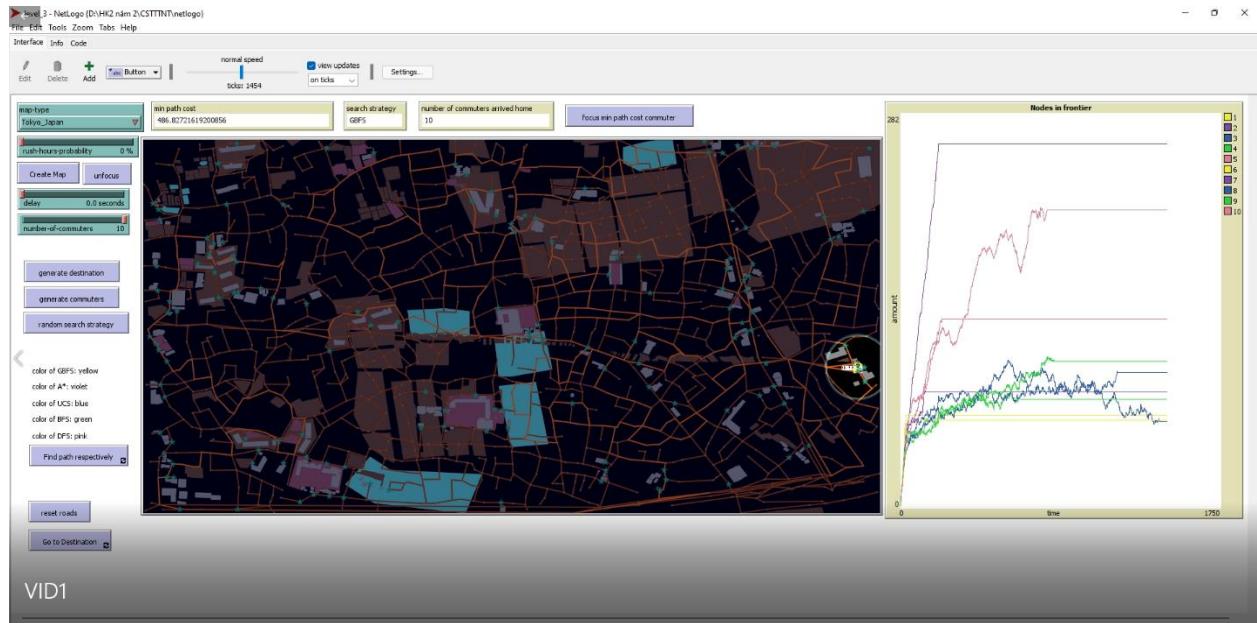
Returned path from end to start using DFS on Kyiv map

+Comments on level 2: Through test cases we can see that DFS and BFS take most of time to run on every map. GBFS, A* and UCS have the run time almost the same but the path returned is different significantly from DFS and BFS. In the map florida, it's

a quite hard challenge for A* and GBFS with large rush hours, it's usually stucks somewhere.

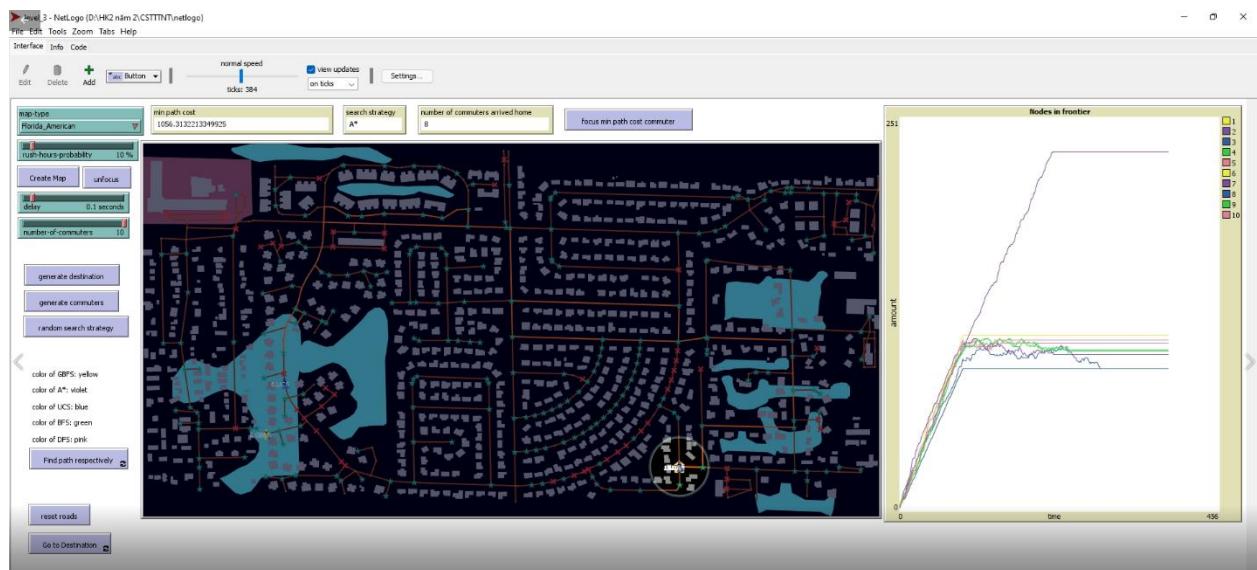
_Level 3:

+Test case 1:



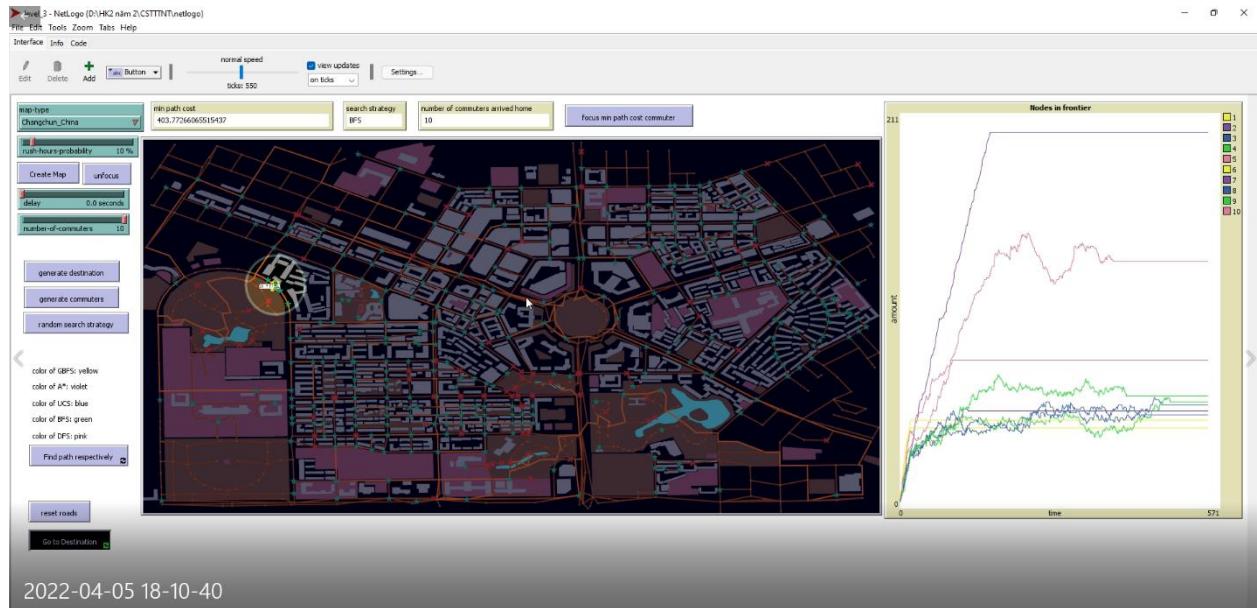
Result after run 5 algorithms on the Tokyo – Japan map

+Test case 2:



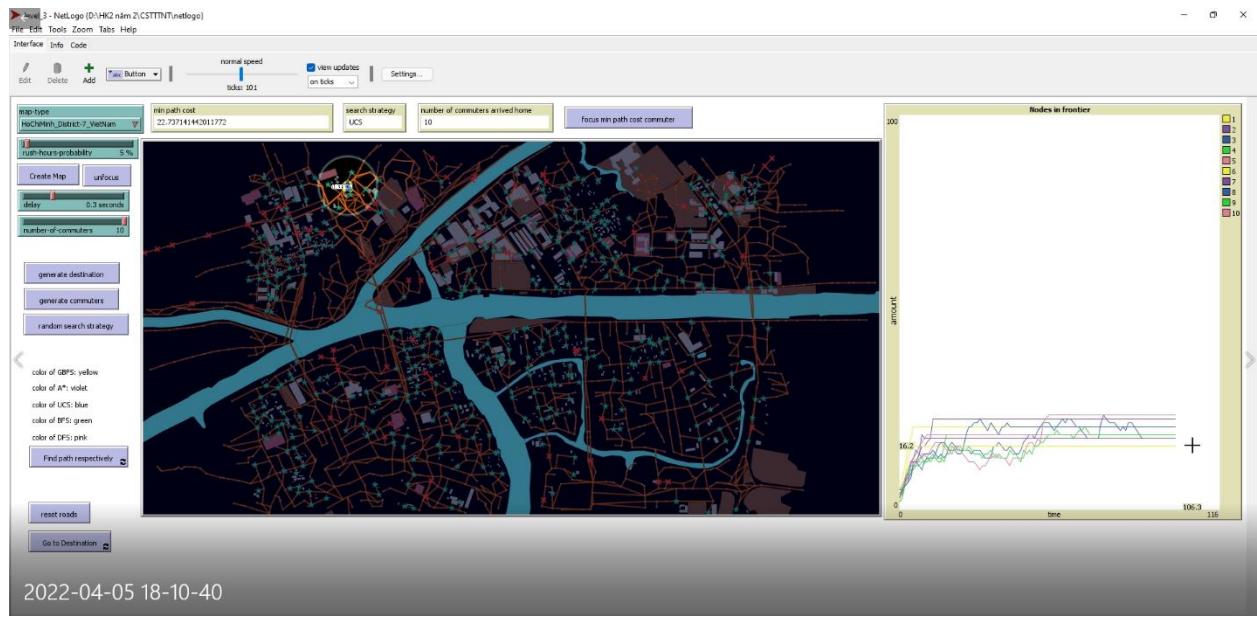
Result after run 5 algorithms on the Florida – America map

+Test case 3:



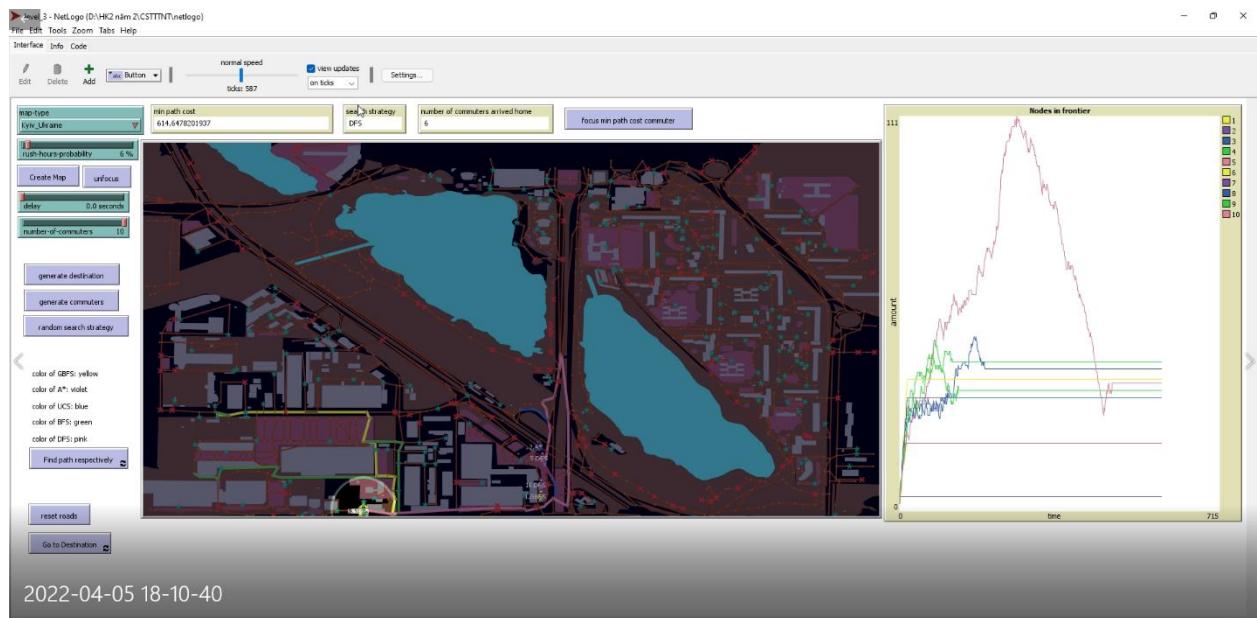
Result after run 5 algorithms on the Changchun – China map

+Test case 4:



Result after run 5 algorithms on the Ho Chi Minh – Vietnam map

+Test case 5:



Result after run 5 algorithms on the Kyiv - Ukraine map

+Comments on level 3: Through five test cases with five different maps, with 5 algorithms run at once, although the locations of the agents are randomized but the distance to goal of agents is the same, different maps with different rush hour probabilities, each test case is generated result for an algorithm has the smallest path cost differently but in the general, GBFS generated the most optimized resolution because it found goal quickest and have the smallest space cost for the frontier.

VII/Videos for requirements:

https://drive.google.com/drive/folders/1D2cX7LCNhO8DSybnni_RRAwM0tTfS43M?usp=sharing

https://www.youtube.com/watch?v=6_sO2b7nt0c

Drive link contains:

_Requirement 1: videos for requirement 1.

_Requirement 2: videos for test cases of 3 levels.

Youtube link for video that merge 3 videos of test cases for 3 levels

VIII/Files in the project:

_”images” folder: Photos about the buttons in the project.

_”map”: folder: Maps used in the project.

- _ "map color" folder: Colors used for agents and environment.
- _ "map_from_statelite": Images about the are we took the map from.
- _ "map_key" folder: Images for us to know which color was used for which algorithm in the project.
- _ "level_1.nlogo", "level_2.nlogo", "level_3.nlogo": code for 3 levels.
- _ "map_key.md", "README.md", "map_key.pdf": files for further explanation.
- _ "report.pdf": report for this project.

IX/References:

<https://ccl.northwestern.edu/netlogo/docs/>

<https://mygeodata.cloud/converter/osm-to-shp>

<https://www.youtube.com/watch?v=NGNCenhcUu4>

<https://github.com/abmgis/abmgis/tree/master/Chapter08-Networks/Models/GMU-Social>

<http://www.cs.us.es/~fsancho/?e=131>

<https://www.youtube.com/watch?v=3CIokwPBDfE>