
Cell2Fire

Release 1.0.0

Jul 13, 2021

Contents

1	Disclaimer	3
1.1	Why Cell2Fire is required?	3
1.2	Introduction	4
1.3	Requirements	4
1.4	Installation	4
1.5	Using Docker	5
1.6	How to Run Simulator	6
1.7	Examples	6
1.8	Editing Weather	7
1.9	Ignition Point	15
1.10	Editing Code Line	17
1.11	Changing the grid	18
1.12	Create Graphics for Forest Fire Model	21
1.13	What are HCELLS (Harvested Cells):	22
1.14	Evaluating a Harvest Plan	25
1.15	Heuristics	26



Cell2Fire is a new cell-based forest and wildland landscape fire spread simulator.

This software is for research use only. There is no warranty of any kind; there is not even the implied warranty of fitness for use.

1.1 Why Cell2Fire is required?

There has been a rapid increase in wildfires over the last few decades. As global warming has affected temperatures, precipitation levels, soil moisture through out the globe we have seen a rapid rise in both the number of and areas burned by wild fires.

Countries such as United States, Canada, Chile, Portugal and Australia have lost critical infrastructure, homes and livelihood for millions of people. Despite a lot of efforts, wildfire still remains a complex problem to simulate.

However from our studies we have narrowed down to two of the most important characteristics of wildfire- a. Rate of Spread. b. Intensity of Spread. both of these are influenced by fuel type, fuel moisture, time of the year, and topographical variables for specified fuel cells.

1.1.1 Previous Models

i.Canadian Forest Fire behavior prediction system (FBP) : It includes empirical fire spread rate model that can be used to predict the rate of spread and intensity of wildfire based on weather, fuel moisture, time of the year etc..

ii.Spatial fire growth model like PROMETHEUS : It is a vector based fire growth simulation model that is based on an adaptation of Huygen's principal of wave propagation. Meaning that the propagation of the fire front is modelled in a fashion similar to a wave, shifting and moving forward continuously in space and time.

iii.FARSITE : It is based on United States Forest Service's BEHAVE fire behavior prediction system as well as Huygen's vector based model.

The two models that best simulate historical fires were FARSITE in the US and PROMETHEUS in Canada.

1.2 Introduction

Cell2Fire is a new cell-based forest fire growth model predictor. It is open-source, and exploits parallelism to support the modeling of fire growth across large spatial and temporal scales. The fire environment is characterized by partitioning the landscape into a large number of homogeneous cells and specifying the fuel, weather, fuel moisture and, topography attributes of each cell. As per Canadian Forest Fire Behavior Prediction (FBP) system the rate of spread and the symmetry of fire within each cell is assumed to be elliptical. The Cell2Fire simulator includes powerful statistical and graphical output along with spatial analysis features. This helps to predict the growth of real and realistic hypothetical fires and comparing our fire growth predictions with the ones produced by the Prometheus Fire Growth Simulator.

This is an on-going project and is expected to grow over time. The original draft of the academic paper is available on [arXiv](#) and an updated version is currently under review.

1.3 Requirements

To simplify the installation process we recommend the use of Unix for the following:

- g++
- Boost (C++)
- Eigen (C++)

The following also need to be installed (We recommend using Anaconda) :

- Python 3.6 and above
- numpy
- pandas
- matplotlib
- seaborn
- tqdm
- opencv
- imread
- networkx

To work on documentation we require:

- Sphinx
- sphinx-rtd-theme

1.4 Installation

If you are not a programmer, these instructions may be difficult. If you get frustrated, jump to the section `docker_section`. Also, Windows users who are not programmers should either install the Unix subsystem, or use docker as described in `docker_section`

First, be sure that boost, eigen and gcc are installed on your computer.

Download the zip file and extract it in a folder in your preferred directory (I used Desktop). Or clone the github repository (these instructions assume you cloned in Desktop)

Give a terminal command to cd to the C++ directory, e.g.,

```
cd Desktop/Cell2Fire/cell2fire/Cell2FireC
```

and edit Makefile to have the correct path to Eigen. Then,


```

1  make
2  cd ../../.. # (the working directory should now be Cell2Fire)
3  pip install -r requirements.txt
4  python setup.py develop

```

1.5 Using Docker

If you have a lot of trouble installing the software on your computer, you can try running it under a docker image. If you are not familiar with docker, you may find it a little confusing, but the installation of Cell2Fire is much simpler (once you have docker installed and running). You also need to install `git`.

Note that in these instructions some command arguments have a single dash, while others have a double-dash.

1.5.1 Install Docker

Install docker and start the docker daemon: <https://docs.docker.com/get-docker/>

1.5.2 Install Cell2Fire in the Docker Image

On a Unix host machine, run this sequence of terminal commands. If any command fails, the subsequent commands will fail:

- `git clone https://github.com/cell2fire/Cell2Fire`
- `cd Cell2Fire`
- `docker run -it --mount source=$(pwd),destination=/Cell2Fire,type=bind dlwoodruff/c2fcondatest:latest`
- `cd Cell2Fire`
- `python setup.py develop`
- `cd cell2fire`
- `cd Cell2FireC`
- `make`
- `cd ..`

To test your installation, try

```
python main.py --help
```

When you are done using the docker image, give the command `exit` to the hashtag prompt.

You don't have repeat the installation steps every time you run. In subsequent sessions you can `cd` to the Cell2Fire directory that you installed and start with the `docker run` command.

1.5.3 Running on Windows

On Windows machines, terminal commands are given to the DOS command prompt. These instructions assume you are running in DOS terminal and not some other shell.

Replace the `docker run` command with this:

```
• docker run --rm -it -v %cd%:/Cell2Fire dlwoodruff/c2fcondatest:latest
```

Note that it might take docker a minute or two to start the image. The commands you give to the docker image (with the # prompt) are unix commands, even if the docker image is running on a Windows machine.

1.5.4 File access

The `-v` argument (or `--mount` on Unix) gives the image (and programs running it) access to the entire Cell2Fire directory structure on your computer, but it will not have access to any other files or directories on your computer. But you want to specify subdirectories, you use the forward slash as in Unix rather than the backslash even if your host computer is a Windows machine.

The commands in the documentation assume you are running on Unix, and if you are running inside the docker image, then you are running on Unix. The docker image has its own Python and it cannot access any other Python.

1.6 How to Run Simulator

In able to run the file you will have to cd to Cell2Fire/cell2fire

then the following command can be used in this directory:

```
python main.py --input-instance-folder ../data/Sub40x40/ --output-folder ../results/  
↳Sub40x40 --ignitions --sim-years 1 --nsims 5 --finalGrid --weather rows --nweathers_  
↳1 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.0 --seed 123 --stats --  
↳allPlots --IgnitionRad 5 --grids --combine
```

This command sends output to the directory specified by `--output-folder`, namely `../results/Sub40x40`.

For the full list of arguments and their explanations use the following command in the same directory:

```
python main.py -h
```

1.7 Examples

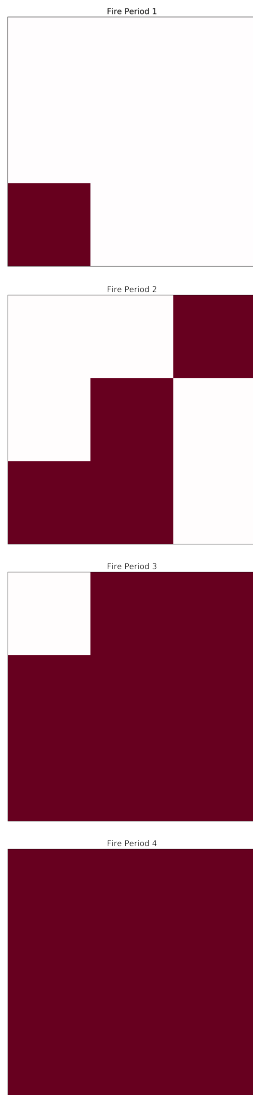
At present, this example assumes you are running on a Unix.

In the contributed folder there is a folder labeled delme63. This example uses a 9 cell forest where you are able to change weather, wind speed, etc. This will provide you results and plots hourly of the fire propagation.

to run this simulation you will have to use the bash command from the Cell2Fire/cell2fire directory. In the command line write the following:

```
bash ../contributed/delme63/go.bash
```

the result output will be hourly stats concluding with the current conditions at 4 hours all 9 cells have been burned. These stats will also be available in your delme63 folder. In delme63/9cellsC1/Plots you will also get hourly plots of the fire propagation.



1.8 Editing Weather

One of the benefits of this simulator is the ability to be able to modify inputs to create different environments. To modify inputs for the delme63 example, you will cd to

```
Cell2Fire/contributed/delme63/9cellsC1
```

From here you will find a file labeled Weather.csv which I recommended using Excel to edit.

In this file the following inputs are editable.

- Temperature [TMP] (Celcius)
- Relativity Humidity [RH]
- Wind Speed [WS] (km/hr)
- Wind Direction [WD] (degrees)
- Fine Fuel moisture Code [FFMC]

- Duff Moisture Code [DMC]
- Drought code [DC]
- Initial Spread Index [ISI]
- Buildup Index [BUI]
- Fire Weather Index [FWI]

In the case of the Buildup Index, Initial Spread Index, and Fire Weather Index these inputs use the values of the other inputs to calculate their respective scores. It is recommended to use the equations found in *Equations for the Canadian Forest Fire Weather Index System* (Van Wagner). This paper clearly explains how generate these values using the other inputs.

1.8.1 Temperature & Humidity

One of the inputs that could be manipulated is the temperature humidity of the environment. Predicting the different outcomes that could occur depending heat and humidity of the region. This gives detailed distinction of how a fire would propagate in hot and humid area, or in a area that is generally colder and with higher amounts of precipitation. Temperature is measured in Celcius while relative humidity is measured as a percentage. Temperature affects the ignition of fires. Warmer temperatures allow for fuels to ignite and burn faster, adding to the rate at which a wildfire spreads. When the humidity is low, meaning that there is a low amount of water vapor in the air, wildfires are more likely to start. The higher the humidity, the less likely the fuel is to dry and ignite.

1.8.2 Wind Speed and Direction

Wind influences heavily how effective fires are and how fast they spread through the forest. A stronger signifies a more intense fire causing a wild fire to spread faster. By default currently we have that the fire starts from the bottom left cell. How Wind direction works and is measured as a degree of where the wind is coming from. For example if a wind is coming from east to west it would coming from 0 degrees but wind coming from West to East would be from 180 degrees. Another quick example would be if you want to input wind coming from NW direction then it would be between 90 and 180 degrees.

1.8.3 Example

To show how modifying these inputs would effect the simulation we will take two different environments and see how fire would spread in these different scenarios.

The first set up is based on a cooler and drier climate with low Humidity with typically high wind speeds. The following inputs were used for this simulation:

- Temperature: from 1pm-5pm was 25 celcius and from 6pm-8pm was 20 celcius. Humidity kept constant at 48%
- Wind Speed: from 1pm-5pm was 23 km/h and slowed down to 6 km/h from 6pm-9pm.
- Wind Direction: Constant at 135 degrees (North Western winds)

The second simulation is based on a area with higher temperatures, humid, and little to no wind. The following inputs were used for this simulation:

- Temperature: from 1pm-5pm was 33 celcius and from 6pm-8pm was 27 celcius. Humidity kept constant at 90%
- Wind Speed: from 1pm-5pm was 5.9 km/h and slowed down to 4 km/h from 6pm-9pm.
- Wind Direction : from 1pm-5pm winds are directed at 30 degrees, from 6pm-8pm wind are directed from 290 degrees.

In the first simulation we see that by the 8th hour that the fire has spread to all of the cells. Higher windspeeds usually influences how fast and effectively fires are able to spread.

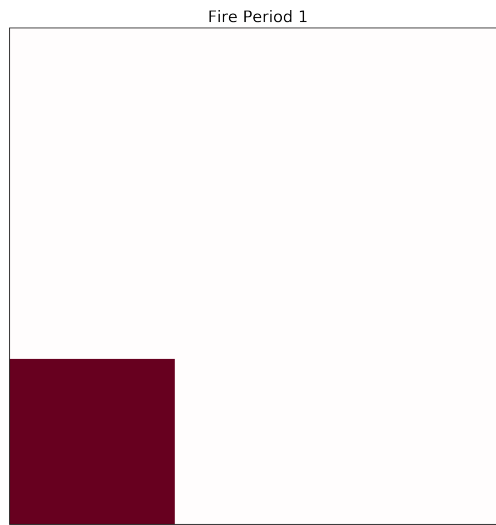


Fig. 1: 1st hour

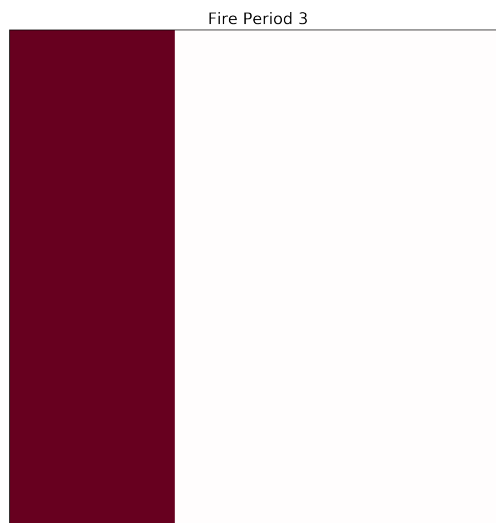


Fig. 2: 4th hour

Notably, in the second scenario we see that fire is not able to spread to all of the cells. This is mostly in part due to the high humidity and the low wind speeds.

1.8.4 Build Up Index

The BUI is a weighted combination of the DMC and DC to indicate the total amount of fuel available for combustion by a moving flame front. The Duff Moisture Code (DMC) indicates the moisture content of loosely-compacted organic layers of moderate depth while the Drought Code (DC) indicates moisture content in deep, compact organic layers. The BUI scale starts at zero and is open ended, a rating above 34 is deemed high and after 77 it is considered extreme.

Editing the BUI input would change how much fuel is there for the fire to spread from the initial cell to the next. The following test has a low BUI value

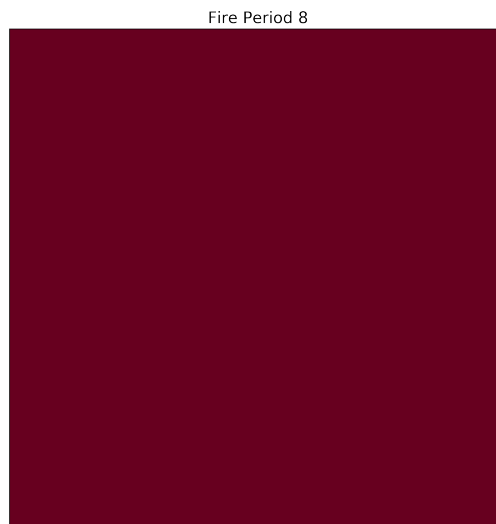


Fig. 3: 8th hour

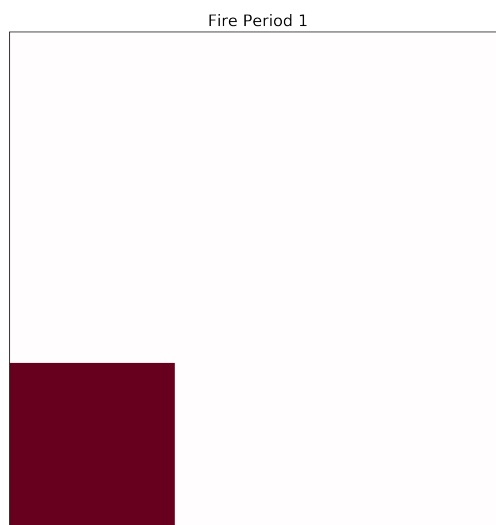


Fig. 4: 1st hour

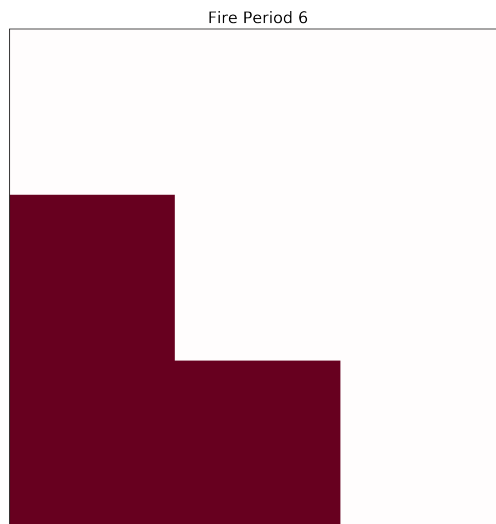


Fig. 5: 6th hour

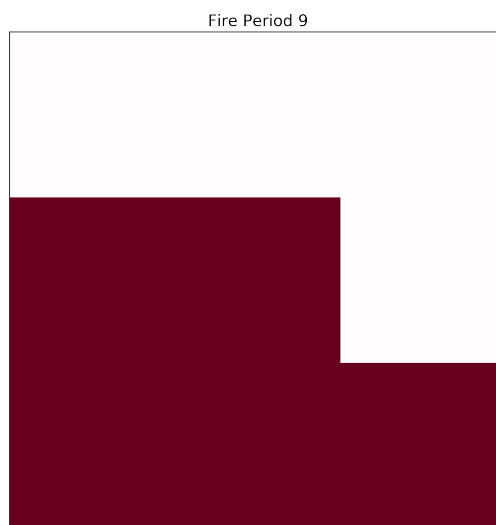
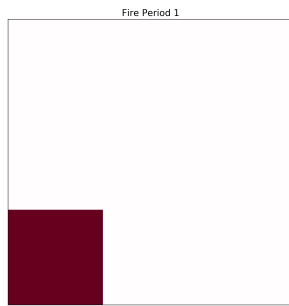
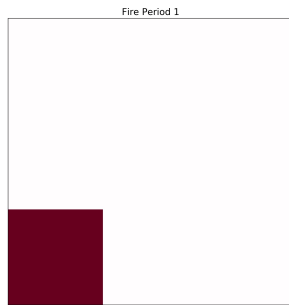
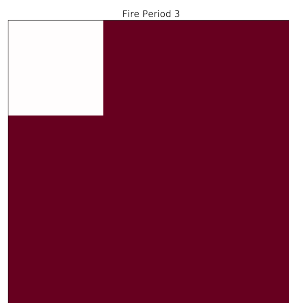
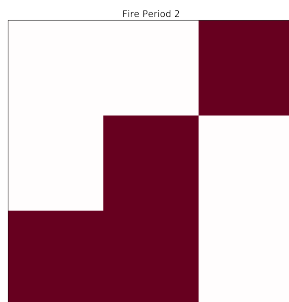
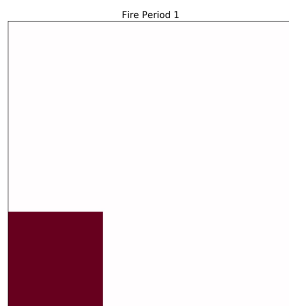


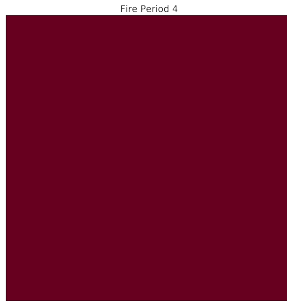
Fig. 6: 9th hour



The fire is kept for a few hours in the same cell and it will never be able to proceed and burn other cells as there is no fuel for the fire.

Test 2 has an extreme BUI score and will be set at 99:





In Test 2 the fire has an extensive amount of fuel and is able to burn all 9 cells in 4 hours. When changing BUI there is a way to calculate a typical value but it is important to know that the DMC value has more weight when getting your value for BUI. It is also important to know that when you have a DMC value of 0 then BUI is zero.

1.8.5 Initial spread Index

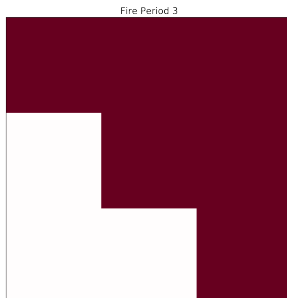
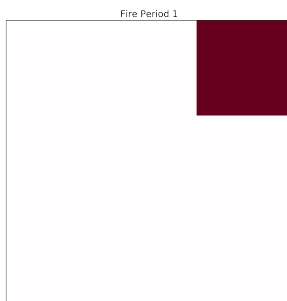
The ISI combines the FFMC and wind speed to indicate the expected rate of fire spread. Generally, a 13 km/h increase in wind speed will double the ISI value. The ISI is accepted as a good indicator of fire spread in open light fuel stands with wind speeds up to 40 km/h.

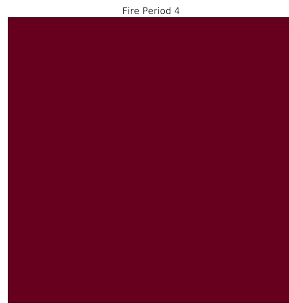
To explain how ISI works we will have to test one with low winds and high a FFMC value

For Test 1 we have :

- Wind speed 4 km/h
- FFMC 95
- ISI is calculated to be 35.9

we get the following plots



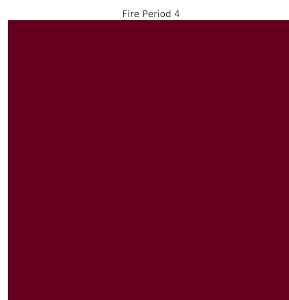
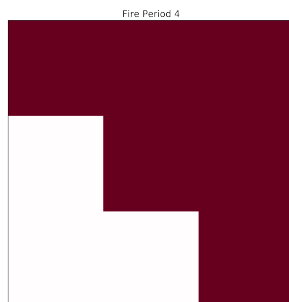
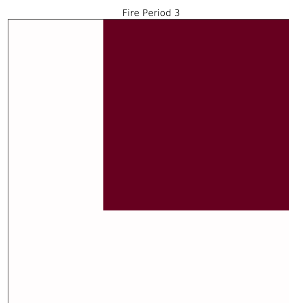


with these values it will take up until the 4th hour to be able to burn all of the cells.

But knowing that a 13 km/h will double the ISI value test 2 will have values :

- Wind speed 17 km/h
- FPMC 95
- ISI is calculated to be 71.8

these inputs resulted in the following graphs :



The first image is at the end of the 1st hour and the final image is at the end of the 3rd hour so we are able to see that increasing the ISI will make the fire spread faster

1.9 Ignition Point

To edit in what cell the fire will start you will have to find your Ignitions.csv file located in

```
Cell2Fire/contributed/delme63/9cellsC1
```

You can edit this file preferably with Excell or Numbers(Mac). The numbering of the cells is the following:

1	2	3
4	5	6
7	8	9

It gives you the options to edit in which cell the fire would start in year 1-4. Where the fire starts carries importance on how effective it spreads with the wind direction.

For example lets take the following cases in consideration.

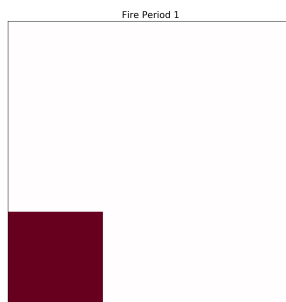
1.9.1 Tests

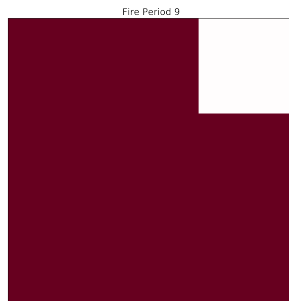
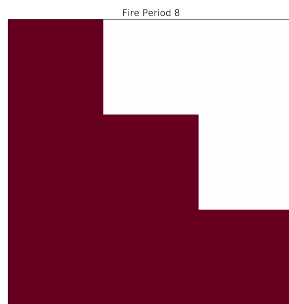
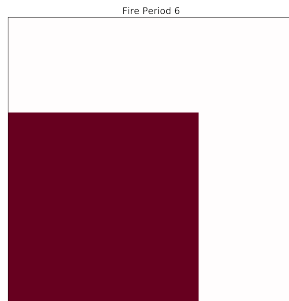
All test will have the following values:

- Wind speed: 25 km/h
- Wind Direction: 45 degrees (NE winds)
- All other values are kept as originally.

What we will change is in which cell the fire starts and see how that effects the effectiveness of the fire.

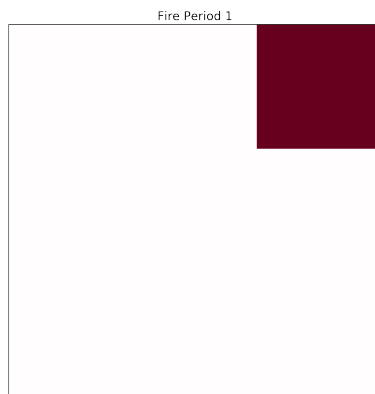
In test 1 we will keep the Ignition point at its original starting cell 7.

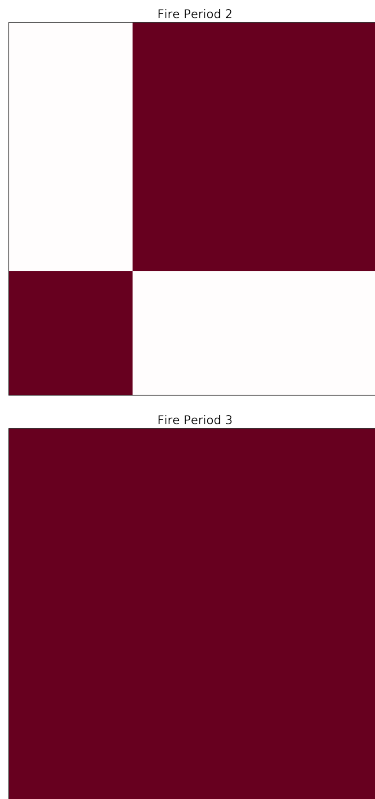




In this case we see that the fire is not able to spread to all of the cells. This due to both the starting position and wind direction as it is harder for the fire to spread against the wind direction.

In Test 2 the Ignition point will be placed in cell 3:





In this test it only takes three hours for the fire to completely cover all the cells. When starting in cell 3 it has the benefit to be able to spread easily with the direction of the wind.

1.10 Editing Code Line

To make change to the code line we first need to be in the right directory. Assuming the file is installed in “**mydirectory**” we open the following path:

```
mydirectory/contributed/delme63/go.bash
```

Once in the right directory we can use any one of the text editing apps (Xcode, Atom etc.) to get the following code line.

```
mydir=../../contributed/delme63
python main.py --input-instance-folder $mydir/9cellsC1/ --output-folder $mydir/results
--ignitions --sim-years 1 --nsims 1 --finalGrid --weather rows --nweathers 1
--Fire-Period-Length 1.0 --output-messages --ROS-CV 0.0 --seed 1134 --stats --allPlots
--IgnitionRad 5 --grids --combine --verbose
```

The path of the working directory should be correctly edited to avoid any confusion. We could get started by editing the following inputs:

- Number of years (Up to 4).
- Number of simulations.

1.10.1 Changing number of years

In the command line you will see a section as shown below:

```
--sim-years 1
```

Here we can enter the number of years and run the code to see the changes in output. This would help in collecting more data with varying number of years.

1.10.2 Number of simulations

We can also change the number of simulations and collect data for the changes caused by number of simulation. This can be edited in the command line as shown below:

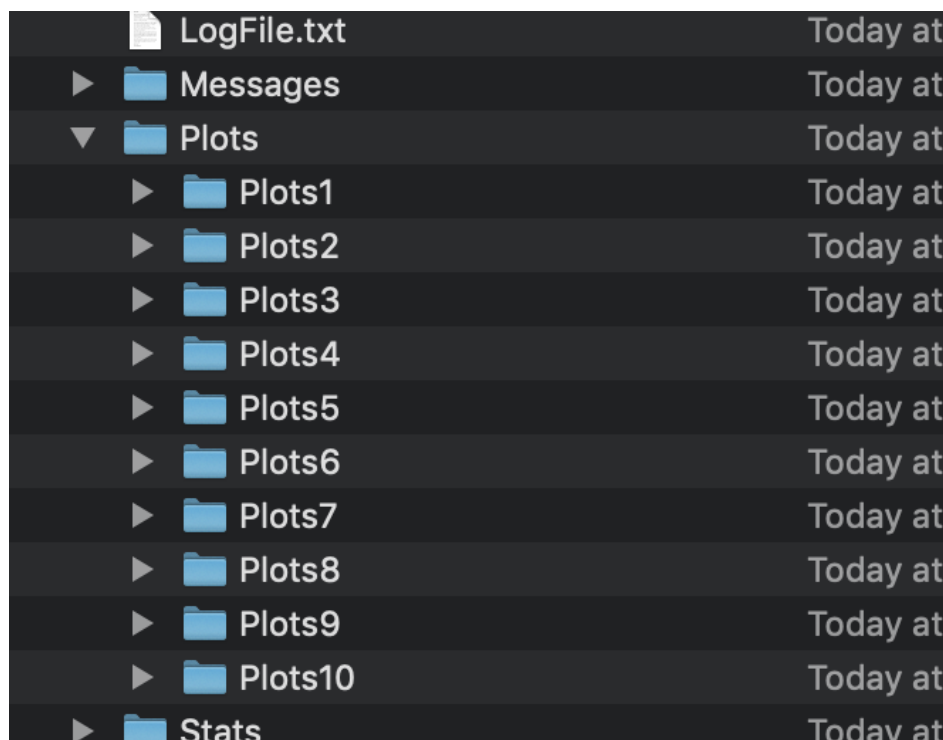
```
--nsmis #
```

This will give you multiple outputs depending on the number of simulations.

If the command line had:

```
--nsmis 10
```

It will result in 10 plots:



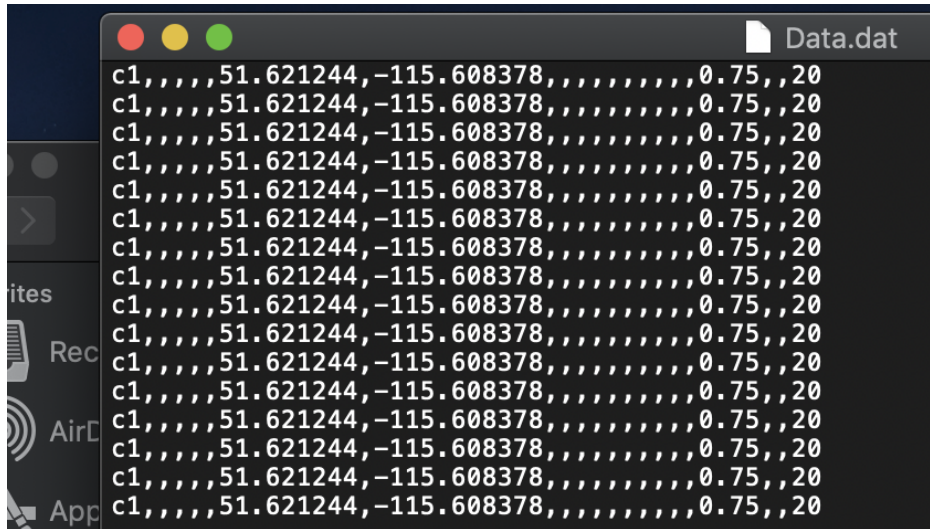
1.11 Changing the grid

The Cell2Fire simulator runs on a variety of grid specifications depending on user expectations. For the flexibility of our software usage, we also provide an option to customize as per user requirements. To customize follow the steps mentioned below:

- Find your contributed folder and create a duplicate of delme63 in it.
- Change the path name so that you are in the right directory. (Look at editing Code Line section for help)

To edit the grid size first open 9cellsC1 folder. In this folder we need to first find Data.csv and Data.dat. We can then open these files either using excel or any other reliable text editor. Once we open these files we can notice that they have 9 data lines each. Here depending on the grid size we require, we could easily copy and paste one of the lines as many number of times as we require to match our grid size. We have presented an example of 16 cell as follows:

Text Editor Snapshot:



Excel Snapshot:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	fueltype	mon	jd	M	jd_min	lat	lon	elev	ffmc	ws	wat	bu	ps	saz	pc	pdf	glt	cur	time
2	C1					51.621244	-115.608378										0.75		20
3	C2					51.621244	-115.608378										0.75		20
4	C3					51.621244	-115.608378										0.75		20
5	C4					51.621244	-115.608378										0.75		20
6	C5					51.621244	-115.608378										0.75		20
7	C6					51.621244	-115.608378										0.75		20
8	C7					51.621244	-115.608378										0.75		20
9	C8					51.621244	-115.608378										0.75		20
10	C9					51.621244	-115.608378										0.75		20
11	C10					51.621244	-115.608378										0.75		20
12	C11					51.621244	-115.608378										0.75		20
13	C12					51.621244	-115.608378										0.75		20
14	C13					51.621244	-115.608378										0.75		20
15	C14					51.621244	-115.608378										0.75		20
16	C15					51.621244	-115.608378										0.75		20
17	C16					51.621244	-115.608378										0.75		20

To create a grid of “n columns” and “m rows” we would require equivalent number of entries as there are rows and columns.

To change cell size we first need to open Forest.asc file present in 9cellsC1 directory. Here we could input the number of rows and columns we require for our own grid. For example to create a 16 cell grid we need to input columns as 4 and rows as 4. An illustration is presented below:

```
Forest.asc
ncols 4
nrows 4
xllcorner 457900
yllcorner 5716800
cellsize 100
NODATA_value -9999
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

At the very end of the code we need to redesign the grid using 1's as per our requirements. Once we have made the desired changes we need to save our file and run the code to get our desired results. The different sized grids are as below:

Image 1:

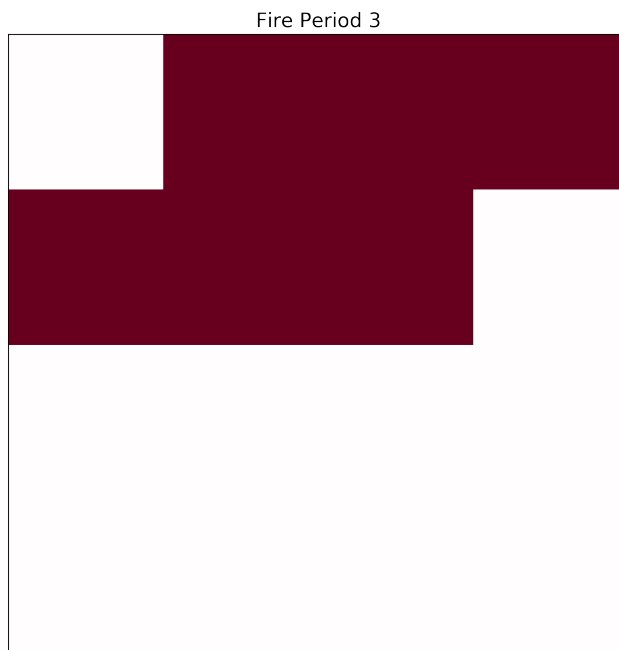
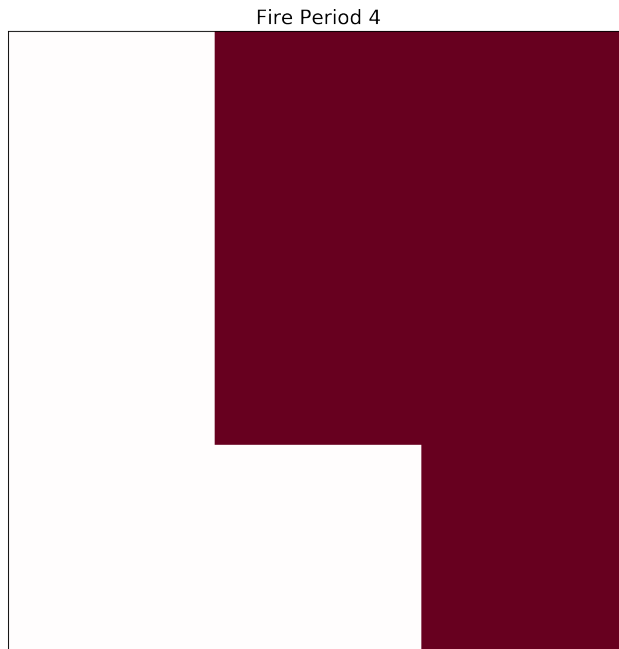


Image 2:



Do not forget to add the number of wanted cells along with respected number of rows and columns.

1.12 Create Graphics for Forest Fire Model

Once we have run our Cell2Fire using docker image we create various outputs in the form of **.png** which are saved in the contributed folder. As the outputs are based on same geographical location with change in time being the only difference, therefore we can easily showcase these images in the form of clips. This would make it simpler to analyze the spread of fire and the magnitude of its extent.

1.12.1 To convert the images in the form of **.gif** files we have to follow these steps:

- We need to download **Pillow** to convert .png files to .gif.

Open the bash window and download using the command

```
1 pip install Pillow
```

- Once we have installed the pillow package we could open our code block gif.py

in the following file location.

```
1 mydirectory/Cell2Fire/docsrc/source/gif.py
```

- In the python program we need to specify the location of the images which would be converted to gif. Once we mention the **.png** files that need to be converted we store the output in the desired location to be used as an output.

This gif.py file can also be used to convert other outputs to the **.gif** format.

1.13 What are HCELLS (Harvested Cells):

The different types of cells in our model are:

- NCells denote the total number of cells in our grid.
- HCells are the harvested cells.

Hcells denote the harvested cells in our grid which can not catch fire and act as breaks to wildfires. These cells could have been harvested either by earlier fires or can strategically be cut to reduce the spread of fire and arrest the spread of fires so that large contiguous areas don't catch fire.

To simulate the program with harvested cells, we need to run the program and set-up the Cell2Fire folder from github. Then we run the program and once the file structure is created in our Cell2Fire document, we could open the Data folder and create a new .csv file to provide input for `-HarvestedCells` command.

1.13.1 How to create a new CSV file

A harvested cells file has comma separated values (it is a .csv file). It should contain the year number and the cell numbers as inputs; e.g. 1, 1, 2, 3, 5, 8 indicates that before year 1, cells 1,2,3,5, and 8 are already harvested. Once we have created the CSV file we could save it as `harvestedCells.csv` in the following path `../data/Harvest40x40`.

Note: The 1st row of the .csv file is not parsed by cell2fire, therefore we could provide “Year number” and “Cell Numbers” in the 1st row. In the second row we can provide our inputs.

Now to simulate the program we would need to provide the inputs required by each function. For `-HarvestedCells` we would have to provide the location of the `harvestedCells.csv` file as an input.

1.13.2 Illustrations 1:

For our 1st illustration we save our `harvestedCells.csv` file with just three harvested cells. We have taken the year number and cell numbers as 1,936,976,1056. Once we save the `harvestedCells.csv` file with the following command:

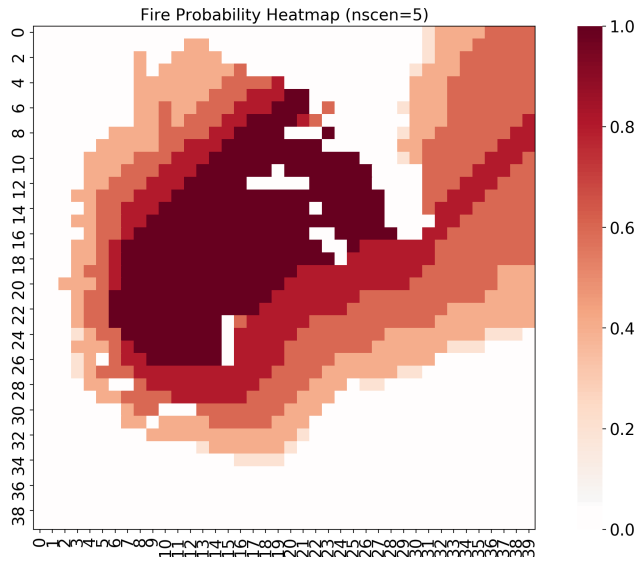
Input Command:

```
python main.py --input-instance-folder ../data/Harvest40x40/ --output-folder ../
↪ Harvest40x40 --ignitions --sim-years 1 --nsims 5 --grids --finalGrid --weather rows_
↪ --nweathers 1 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.8 --seed 123 --
↪ stats --allPlots --IgnitionRad 1 --grids --combine --heuristic 1 --GASelection --
↪ HarvestedCells ../data/Harvest40x40/harvestedCells.csv
```

Output:

Once we run the program we create a series of outputs in `Harvest40x40` folder which would be saved in the `Cell2Fire` directory. The `Harvest40x40` folder will have the output in the form of Grids, Plots and Stats.

We could show how the fire has spread through the `BP_HeatMap.png` saved in the `Stats` Directory.



We can see as the harvested cells are very few, we can not fully stop the propagation of fire. It spreads and covers the entire fire area.

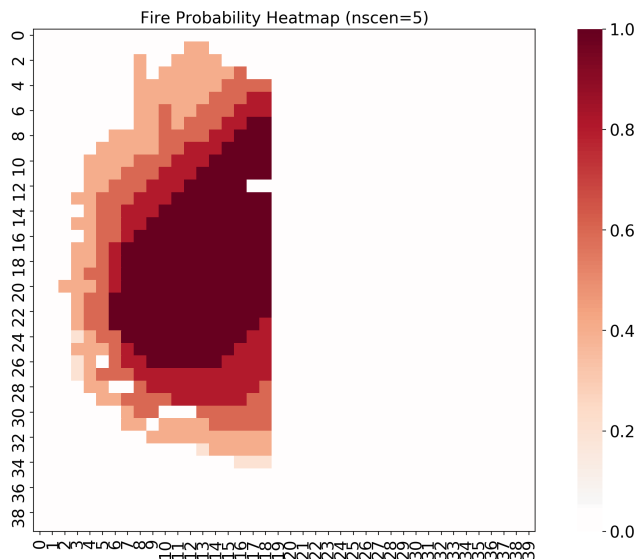
1.13.3 Illustrations 2:

```
python main.py -input-instance-folder ../data/Harvest40x40/ -output-folder ../Harvest40x40 -ignitions
-sim-years 1 -nsims 5 -grids -finalGrid -weather rows -nweathers 1 -Fire-Period-Length 1.0 -output-
messages -ROS-CV 0.8 -seed 123 -stats -allPlots -IgnitionRad 1 -grids -combine -heuristic 1 -GASe-
lection -HarvestedCells ../data/Harvest40x40/harvestedCells.csv
```

Output:

Once we run the program we create a series of outputs in Harvest40x40 folder which would be saved in the Cell2Fire directory. The Harvest40x40 folder will have the output in the form of Grids, Plots and Stats.

We could show how the fire has spread through the BP_HeatMap.png saved in the Stats Directory.



As we have harvested enough cells the fire does not propagate. We have strategically harvested cells in a straight line starting from cell 20,60,100..1580. This results in stopping fire spread even though there is more forest cover which

would be burnt if we did not stop its propagation.

1.13.4 Illustrations 3:

For our 3rd illustration we save our harvestedCells.csv file with multiple harvested cell (1,42,83,124...,16000). We have harvested the forest diagonally to see how the fire propagates. We have take the year number and cell number as 1,1,42,83,124...,1600. Once we save the harvestedCells.csv file with the following inputs we can parse them using our program. The command for our program with their respective inputs is as shown below.

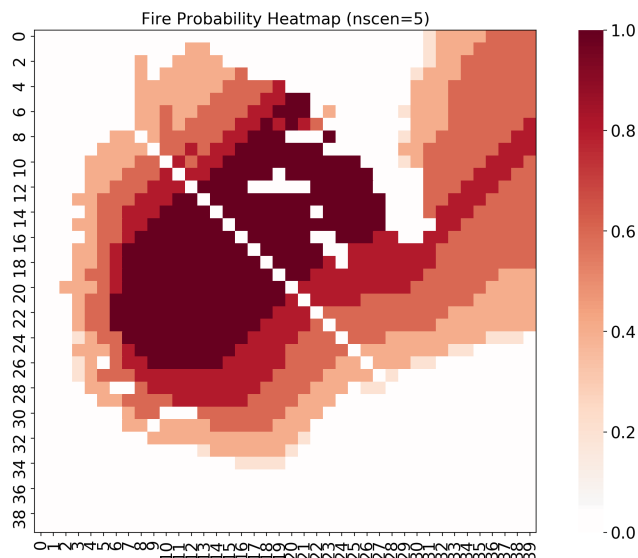
Input Command:

```
python main.py --input-instance-folder ../data/Harvest40x40/ --output-folder ../
Harvest40x40 --ignitions --sim-years 1 --nsims 5 --grids --finalGrid --weather rows
--nweathers 1 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.8 --seed 123 --
stats --allPlots --IgnitionRad 1 --grids --combine --heuristic 1 --GASelection --
HarvestedCells ../data/Harvest40x40/harvestedCells.csv
```

Output:

Once we run the program we create a series of outputs in Harvest40x40 folder which would be saved in the Cell2Fire directory. The Harvest40x40 folder will have the output in the form of Grids, Plots and Stats.

We could show how the fire has spread through the BP_HeatMap.png saved in our Stats Directory.



As we can see even though we have harvested the cells in a diagonal the fire still spreads. This particular example would help to strategically decide on how to harvest the forest. It also shows that the fire would spread even if the adjacent cells are in contact with the burning cells with just their corners in contact with each other. Therefore to completely arrest spread of fire there should be no contact between forested cells and the ones having an active fire.

Application:

The output of the program would be stored in Cell2Fire document. We can access the new folder that would be created after running the program and access the various Plots, Stats, Grids, and messages.

By running multiple simulations we could reach an optimal solution which would enhance our chance to utilize our model to effectively manage the spread of forest fires.

1.14 Evaluating a Harvest Plan

Harvest plans are entered in csv file that has one row for each year with the year number followed by the cells harvested at the beginning of that year. The `--HarvestedCells` option gives the name

1.14.1 Band first year:

This is very artificial harvest plan to illustrate how the inputs work. For the first have take the year number and cell numbers as 1,20,60,100,140...,1580. The second year we plan to harvest a cell next to the cell harvest the previous year (alternating sides), so the input for the second year is 2, 19,61,99,141 ...

Input Command:

```
python main.py --input-instance-folder ../data/Harvest40x40/ --output-folder ../
↪Harvest40x40 --ignitions --sim-years 2 --nsims 1 --grids --finalGrid --weather rows_
↪--nweathers 1 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.0 --seed 123 --
↪stats --allPlots --IgnitionRad 1 --grids --combine --HarvestedCells ../data/
↪Harvest40x40/band1_2.csv
```

Output:

Once we run the program we create a series of outputs in the Harvest40x40 folder. The Harvest40x40 folder will have the output in the form of Grids, Plots and Stats.

As we have harvested enough cells the fire does not propagate. We have strategically harvested cells in a straight line starting from cell 20,60,100..1580. This results in stopping fire spread even though there is more forest cover which would be burned if we did not stop its propagation via harvesting. So there should be some

cells harvested in the second year.

1.14.2 Stochastics?

ROS CV

In the next command we allow variation in the rate of spread by giving it a CV of 0.8 so we run 5 simulations to get statistics (which won't vary much).

```
python main.py --input-instance-folder ../data/Harvest40x40/ --output-folder ../
↪Harvest40x40 --ignitions --sim-years 2 --nsims 5 --finalGrid --weather rows --
↪nweathers 1 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.8 --seed 123 --
↪stats --allPlots --IgnitionRad 1 --grids --combine --HarvestedCells ../data/
↪Harvest40x40/band1_2.csv
```

Random Ignitions

In the next command, we add uniform random ignitions (but dropping the `--ignitions` option).

```
python main.py --input-instance-folder ../data/Harvest40x40/ --output-folder ../
↪Harvest40x40 --sim-years 2 --nsims 5 --grids --finalGrid --weather rows --nweathers_
↪1 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.8 --seed 123 --stats --
↪allPlots --IgnitionRad 1 --combine --HarvestedCells ../data/Harvest40x40/band1_2.csv
```

Random Weather

Random weather is supported by providing multiple weather files in the `Weathers` sub-directory of the input directory. Each weather file has the name `WeatherX.csv`, where `X` is replaced by an integer. There should be one file per integer (`0..nweathers`) and use `--weather random` to trigger randomly selecting one for each run (note whether random ignitions is on or off).

```
python main.py --input-instance-folder ../data/Harvest40x40/ --output-folder ../
↪Harvest40x40 --ignitions --sim-years 2 --nsims 5 --grids --finalGrid --weather_
↪random --nweathers 200 --Fire-Period-Length 1.0 --output-messages --ROS-CV 0.8 --
↪seed 123 --stats --allPlots --IgnitionRad 1 --combine --HarvestedCells ../data/
↪Harvest40x40/band1_2.csv
```

The value of `--nweathers` should be less than or equal to the number of the weather files in the `Weathers` subdirectory.

1.15 Heuristics

Cell2Fire includes built-in heuristics to use as benchmarks.

1.15.1 A Simple Example

First, move to the directory where “main.py” is located. From the root directory of Cell2Fire, you can `cd` into “cell2fire”:

```
cd cell2fire
```

Then, run the following command:

```
python main.py --input-instance-folder ../data/Sub20x20/ --output-folder ../results/
↪Sub20x20/Sub20_RW_RI_N10 --sim-years 1 --nsims 10 --finalGrid --weather random --
↪nweathers 100 --Fire-Period-Length 1.0 --ROS-CV 0.0 --seed 123 --IgnitionRad 0 --
↪stats --output-messages --ROS-Threshold 0 --HFI-Threshold 0 --heuristic 1
```

Output:

```
cell2fire_path ~/workspace/Cell2Fire/Cell2Fire/cell2fire
End of Cell2FireC execution...
----- Generating Statistics -----
General stats...
----- Generating outputs for heuristics -----
Reading data...
Using custom value function ( from file ../data/Sub40x40/values40x40.csv )
----- Running Heuristic: Max_Utility -----
Total Available cells: 1444
Running the AS-IS forest (no heuristic applied)
End of Cell2FireC execution...

Treat Fraction 0.05...
Adjacent Constraint: False
Demand satisfied: True
Total fitness (FPV): 92733.0
Running the instance with the heuristic...
End of Cell2FireC with Harvesting Plan execution...
```

(continues on next page)

(continued from previous page)

```

Generating stats from heuristic...
General stats...

Treat Fraction 0.1...
Adjacent Constraint: False
Demand satisfied: True
Total fitness (FPV): 177698.0
Running the instance with the heuristic...
End of Cell2FireC with Harvesting Plan execution...
Generating stats from heuristic...
General stats...

....

```

1.15.2 A List of Heuristics

Currently, the heuristics (greedy based) are executed for a hard-coded interval of treatment fraction, i.e., varying the total number of nodes to be treated by 5%, starting from 0% (no treatment) to 90% of the available cells. If a valid argument is provided, Cell2Fire simulates `nsim` fires, gathers statistics and useful information for the metrics, simulates `nsim` fires with the treatment plan provided by the heuristic, and finally, provides an evaluation of the treatment plan's performance. In parenthesis the number of the heuristic associated with the `-heuristic` argument (`-heuristic -1` indicates that no heuristic is applied).

- Random (0): Selects available cells at random until the treatment fraction threshold is hit (e.g., 10% of the available cells)
- Random_Adj (1): Idem as above but satisfying adjacency constraints by selecting cells at random connected to the previous ones.
- Max_Utility (2): Given a utility map with a value for each cell (.csv file, see `customValue` argument), it selects those cells that maximize the total utility.
- Max_Utility_Adj (3): Idem but satisfying adjacency constraints (greedy)
- Burnt_Probability (4): Selects those cells with higher burn probability based on the previous simulations.
- Burnt_Probability_Adj (5): Idem but satisfying adjacency constraints (greedy)
- FPV_Palma (6): Calculates the fire protection value from Palma et. al and selects those cells that maximize the total sum of the selection (slow, not recommended)
- FPV_Palma_Adj (7): Idem but satisfying adjacency constraints (greedy)
- DPV_VaR_Volume (8): Calculates the downstream protection value metric using the volume/provided utility per cell as the VaR to protect. Selects the cells of the forest that maximize the total summation of the metric.
- DPV_VaR_Volume_Adj (9): Idem but satisfying adjacency constraints (greedy)
- DPV_VaR_Volume_Degree (10): Idem as (8) but weighting the metric by the degree of the node.
- DPV_VaR_Volume_Degree_Adj (11): Idem but satisfying adjacency constraints (greedy)
- DPV_VaR_Volume_Degree_Time (12): Experimental. Adds a time factor weighting the metric by the ROS of the fire.
- DPV_VaR_Volume_Degree_Time_Adj (13): Idem but satisfying adjacency constraints (greedy)
- DPV_VaR_Volume_Degree_Time_Layer_decay (14): Experimental. Adds a decay factor associated with how deep the node is to the ignition point. The farther, the smaller importance of the cell.

- DPV_VaR_Volume_Degree_Time_Layer_decay_Adj (15): Idem but satisfying adjacency constraints (greedy)
- BCentrality (18): Calculates the betweenness centrality value for each node, selecting those that maximize the total summation.
- BCentrality_Adj (19): Idem but satisfying adjacency constraints (greedy)

1.15.3 Output Folders

The output directory, specified by the user when executing the command, will have the base case without any heuristics. Some heuristics use that (e.g., bp maps bases hours) to calculate protection metrics.

Then, the heuristic(s) are applied and generates a directory per heuristic in the user-specified output folder.

To access the heuristic results, move to the Heuristic folder. Inside, there is a folder that has the name of the heuristic applied (refer to the list of heuristics in the previous section). For example, if you apply heuristic 2, the name of that directory will be “Max_Utility”. Inside that directory, you can find directories that contain the results created by the heuristic - named as FractionX, where X is the treatment/harvesting intensity.

For instance, if you run the following command:

```
python main.py --input-instance-folder ../data/Sub40x40/ --output-folder ../results/  
↳Sub40x40_H2 --sim-years 2 --nsims 10 --finalGrid --weather random --nweathers 100 --  
↳Fire-Period-Length 1.0 --ROS-CV 0.0 --seed 123 --IgnitionRad 0 --stats --output-  
↳messages --ROS-Threshold 0 --HFI-Threshold 0 --heuristic 2 --customValue="../data/  
↳Sub40x40/values40x40.csv"
```

The results of Heuristic 2 is stored in “results/Sub40x40_H2/Heuristic/Max_Utility”.