Modeling the Spread of Wildfire

Jacob T. Lougee

IESA, University of North Georgia

GISC 3200K

Dr. Huidae Cho

December 4, 2020

Abstract

In wildland firefighting it is useful to know how a fire might spread in order to mitigate risk to the firefighters and prioritize what areas of the wildfire to attack. As GIS analysts who understand remote sensing, we have the ability to measure many of the key determining factors for how a fire will spread such as terrain, ground cover, and plant "wetness" and could even collect data from wildland firefighters in the field and interpolate between them to get a big picture of conditions on the ground. We should, therefore, be able to simulate with some degree of accuracy how a fire will spread by assigning a percent chance of ignition to each pixel in a raster. Depending on the needs of the model we could design a very simple solution that doesn't account for many variables but would be easier to develop, or a very complex solution that could achieve a very high degree of accuracy, constantly update variables, and even start to predict how the input variables will interact and change over time. This paper will explore current wildfire forecasting and modeling techniques, and applications, and, as a proof of concept, demonstrate simulating a hypothetical fire in python with a cellular automata model that is easier to develop, and highly flexible in exchange for some degree of accuracy.

Introduction

Current Literature

A multitude of programs and tools already exist for analyzing data, and forecasting how a particular wildfire might spread. The methods utilized, and the data required to run each model is extremely varied. Current models have many strengths and weaknesses . In wildland firefighting and modeling of fire, there are a wide variety of needs and often underutilized data. For example the Kestrel fire weather unit measures fine dead fuel moisture, probability of ignition, dew point temperature, heat stress index, current wind speed , wind direction, crosswind, temperature, barometric pressure, pressure trend, altitude and density altitude, these units however are mostly used by individual firefighters to help predict the behavior of a fire, instead of for collecting data to get a a big picture of what might happen to the fire.

Jiang, W et al, in their 2020 article, conceptualized a model very similar to the one proposed in this paper. They explained that, "cellular automata models are gaining momentum in recent years since their simple structure and low computational complexity when simulating forest fires." Their paper focused more specifically on modeling fires at the wildland urban interface, where manmade structures and homes have a significant impact on the way that a fire spreads. That work is certainly useful, but adds data requirements, namely data about the location and characteristics of manmade structures, to the model, which treats them as a separate type of cell in the automata model. In order to enable the model that I have proposed to remain flexible, I would suggest instead experimenting with using such valuable datasets to find a formula that can adjust the flammability/chance of ignition in the initial raster so that houses take longer to burn than wild land areas, in a way that at least mostly matches with reality.

In, " A GIS-based fire spread simulator integrating a simplified physical wildland fire model and a wind field model" The authors explain that among numerous wildfire models which have already been developed they, "can be classified into different types according to the nature of their construction: physical, semi-physical or empirical," and that, "although the nomenclature varies. Some authors argue that for a model to be classified as physical it must cater for both the physics and chemistry of the fire spread; semi-physical models are defined as those that seek to represent only the physics of the problem; and empirical models are those based on a phenomenological description or statistical regression

of observed fire behaviour. According to the physical process modelled, wildland fire models can also be grouped as ground fires, surface fires, crown fires and spotting models." (Prieto Herráez, D et al., 2007) By those classifications, the model I have proposed would best be described as an empirical model, however, while it doesn't forecast how each individual variable will change over time, its ability to run given a single initial value for each cell, allowing the user to determine how that value is calculated, means that the results could probably come close, at least for the first few "generations" to the predictions of some physical, or semi physical models.

Prieto Herráez, D et. al. briefly mentioned Rothermel's model (1972), which could have been worth looking into further while writing this program, however the plan for the proof of concept here, is to generate a raster full of random values and to achieve a result that visually matches the reality of how a wildfire might spread. It's possible that any formula which is traditionally used to accurately depict a fire spreading might not achieve such a result when the value from one cell to another is completely random and follows no gradual trends whatsoever. Also, Rothermel's model and the model developed by the authors of "A GIS-based fire spread simulator integrating a simplified physical wildland fire model and a wind field model," require very specific input perimeters that would severely limit the flexibility of this program. That being said, the benefit of using GIS is discussed, namely that it allows for near-real time updates to the model, and that they can run a directional wind model separately or in combination with the fire-spread model, which would be useful for the model I have proposed.

Much like Jiang, W et al did in the first paper I discussed, the authors of, "Wildland Fire Spread Modeling Using Convolutional Neural Networks," considered the need to model wildfires in a way that is not as computationally costly as many of the current models, but instead of using a cellular automata model, they decided to take a machine learning approach. They made some useful notes for developing any wildfire forecasting model. "The impact of wind on wildland fire spread generally follows a power law where the exponent and pre-factor are related to the fuel type. The impact of fuel moisture content on wildland fire spread is generally modeled as strongly linear or weakly exponential based on the fuel type." They did an excellent job of summarizing Rothermel's model, noting that it, "uses empirical correlations for heat source and sink terms [1]. The baseline rate of spread (with no wind or slope) is based on fuel density, type, and moisture content. The impact of wind and slope is modeled as a multiplier of the baseline rate of spread. Although developed to determine the rate of spread from a single ignition source in a single direction, these models have been expanded to work in two-dimensions using Huygens' principle" These notes could prove useful for developing accurate pre-processing algorithms given different input data for my model instead of relying purely on user intuition and experimentation. Based on their work I believe it also may be possible to feed a CNN different sample data sets and outcomes to develop the best possible pre-processing algorithm for a wide variety of input sets for my cellular automata model, and store a library of the best options that users could pick from.

In their paper titled "High-Resolution Rapid Refresh Model Data Analytics Derived on the Open Science Grid to Assist Wildland Fire Weather Assessment," Brian k. Blaylock John d. Horel and Chris Galli discussed a novel approach for the processing, analysis, and storage of large atmospheric datasets, which they acknowledge is useful for flagging outlier wind and weather conditions that would be of primary concern to wildland firefighters and useful for testing wildland fire models. Upon close inspection there isn't much they discussed that I could use for my very simplistic model, but their assertion that, "continued and accelerated production

of data by [big data] sources and forthcoming technologies poses data management and data analysis challenges common to other disciplines," does give demonstrate that there is a niche for such a simplified model that can account for practically any input.

Ying Xie, and Minggang Peng in their 2017 article wrote about the use of ensemble learning, basically assembling multiple learners to solve a statistical problem, and touted the benefits of random forests in predicting and modeling wildfires. Although machine learning has already come up while discussing a previously mentioned paper, it's worth noting that ensemble learning outperformed the standar machine learning and that they were able to test the algorithms on forest fire datasets in the UCI machine learning repository, as it could prove useful for testing pre-processing algorithms for the model proposed in this paper on a some of that data if one could publicly access it.

The developers of IRIS, a rapid response fire forecast model, had similar goals in mind to the cellular automata model proposed here, and much of what they mention is relevant. For one they assert that while, "fire models range from tools such as BehavePlus and FARSITE, which are based on the well-documented fire spread equations of Rothermel, to advanced 3D computational fluid dynamics and combustion simulation models, such as FIRETEC and WFDS, each model has its advantages and disadvantages which depend primarily on the computational cost, data requirements, accuracy, robustness and transferability" (Giannaros, T. M, 2019). And to their point, even their own model is very limited, requiring as inputs; a mesoscale analysis and forecast, high resolution topography and land use data, and ultra high resolution (100m) fuel and topographic data, while at the same time, the model does not benefit from additional input data.

Finally Page et. al in their article from 2017 Evaluated NDFD weather forcast as a model input for forecasting the behavior of wildland fires because, "Wildland fire managers in the United States currently utilize the gridded forecasts from the National Digital Forecast Database (NDFD)". They found that the forecasts were generally reliable except that they under-predicted higher wind speeds, which is quite useful to know for anyone modeling wildfires, and also gave some potentially useful information about the behavior of wildfires in general. Of note is that, "moisture tends to dampen fire spread as a result of the high specific heat of water, and the local wind field enhances forward fire spread and intensity by increasing both the rate of combustion and by directing hot combustion products toward unburned fuels." They also say, "Near-surface wind speed and direction are affected by terrain and vegetation through mechanisms such as channeling or sheltering. Local and large-scale variations in terrain shape, orientation, and complexity can result in wind flows through valleys that can override and/or enhance synoptic winds. Likewise, vegetation type and size can alter the magnitude of the wind flow near the surface as a result of the effects of bulk drag from crown foliage." That information demonstrates the usefulness of a model like the one proposed for this paper that can use almost any data source, such as a raster interpolated from point data collected by firefighters on the ground for example, to potentially increase its accuracy.

Materials and methods

For the demonstration, no real world data is necessary. The code creates a raster-like board in python with random values assigned to each cell. This is not a realistic representation but does demonstrate "fire" spreading across a grid where some cells ignite faster than others as

a proof of concept. Expressing chance of ignition as a percentage allows for a variety of inputs tailored to what data is currently available to the user.

Results

The demonstration shows un-burnt areas as a blank space and burning or burnt areas as an, “X”. The python script assigns random values to the cells on a board, or in other words, a raster, and shows any cell with a “chance of ignition value” of greater than 99 as a burning or burnt cell. The code checks if any adjacent cells return as “True” (have a value greater than 99) and increases the chance of ignition of the un-burnt cells adjacent to cells that are on fire. Once a non-burning cell reaches a value of 100 or greater it is considered fire and remains at 100. Anything below 25, in this hypothetical, represents an area that could only burn at very extreme temperatures, such as roads, areas with little flammable ground cover, or low lying water saturated earth, and will remain un-burnt by the wildfire. Because this is a proof of concept, it is set up to assign a random value to each cell in the grid and the user defines the size of the raster (or grid). I recommended 20x70 because it fills the default command module window. Successful testing showed a mostly blank grid that fit the default size of the command module, and a small number of cells were set "on fire", represented by an "X" in the initial state. Pressing "Enter" repeatedly added a new "generation" of the cellular model where the fire had spread out from the original points of ignition. It looks like a reasonably accurate approximation of a fire spreading across a map for about 40-60 new generations (at that point, most of the flammable materials on a 20x70 grid have usually already burned).

Conclusions

This python code was a successful test and proof of concept for modeling wildfire with a cellular automata model. It would, however, be more intuitive to use in GIS instead of the command module, which is a future goal for this work. In GIS, any available inputs will determine the initial, “chance of ignition” values in the grid after some pre-processing. This allows for a significant degree of flexibility because that value can be calculated given whatever data is currently available to the analyst. For example, if the analyst only has "dryness" for each cell in the grid (calculated from landsat imagery), they can run the model even with that very limited dataset, but they could also combine multiple layers, like perhaps (1/2*dryness percentage) + (1/2*slope percentage[because heat rises]) and the model would still hold somewhat true. It would also be useful to include this modeling tool in a toolbox with an open-sourced library of preprocessing tools for any combination of datasets. Resampling, for example, would be necessary in most cases in order to start with a pre-determined pixel resolution and a pre determined time interval for each new generation of the raster grid. This way we will be able to incorporate existing scientific understanding about the velocity (distance/time) of wildfires. It is noteworthy however, that this method of modeling the spread of a fire trades some accuracy for flexibility. For example, the model in its current state would be able to utilize windspeed data to increase the likelihood of ignition for a given cell, but would not be able to consider the direction. One future goal should certainly be to add a place for a directional raster as an input as well. This way, a user could have the option to include the direction of several factors such as a slope, and the wind, to significantly increase the accuracy of the model. Because of the limitations to the accuracy of this model, and given its uncommon ability to utilize almost any

information that is available, it would be less useful for long-term forecasting and planning, but more useful for fast decision making and responses to sudden changes in conditions.

References

Agranat, V., & Perminov, V. (2020). Mathematical modeling of wildland fire initiation and spread. Environmental Modelling and Software, 125. https://doi-org.proxygsu-nga1.galileo.usg.edu/10.1016/j.envsoft.2020.104640

Blaylock, B. K., Horel, J. D., & Galli, C. (2018). High-Resolution Rapid Refresh Model Data Analytics Derived on the Open Science Grid to Assist Wildland Fire Weather Assessment. Journal of Atmospheric & Oceanic Technology, 35(11), 2213–2227. https://doi.org/10.1175/JTECH-D-18-0073.1

Cruz, M. G., Alexander, M. E., Sullivan, A. L., Gould, J. S., & Kilinc, M. (2018). Assessing improvements in models used to operationally predict wildland fire rate of spread. Environmental Modelling and Software, 105, 54–63. https://doi-org.proxygsu-nga1.galileo.usg.edu/10.1016/j.envsoft.2018.03.027

Giannaros, T. M., Kotroni, V., & Lagouvardos, K. (2019). IRIS – Rapid response fire spread forecasting system: Development, calibration and evaluation. Agricultural and Forest Meteorology, 279. https://doi.org/10.1016/j.agrformet.2019.107745

Hodges, J. L., & Lattimer, B. Y. (2019). Wildland Fire Spread Modeling Using Convolutional Neural Networks. Fire Technology, 55(6), 2115–2142. https://doi.org/10.1007/s10694-019-00846-4

Jiang, W., Wang, F., Fang, L., Zheng, X., Qiao, X., Li, Z., & Meng, Q. (2020). Modelling of Wildland-Urban Interface Fire Spread with the Heterogeneous Cellular Automata Model. Environmental Modelling and Software. https://doi.org/10.1016/j.envsoft.2020.104895

Linn, R. R., Goodrick, S. L., Brambilla, S., Brown, M. J., Middleton, R. S., O'Brien, J. J., & Hiers, J. K. (2020). QUIC-fire: A fast-running simulation tool for prescribed fire planning. Environmental Modelling and Software, 125. https://doi-org.proxygsu-nga1.galileo.usg.edu/10.1016/j.envsoft.2019.104616

Page, W. G., Wagenbrenner, N. S., Butler, B. W., Forthofer, J. M., & Gibson, C. (2018). An Evaluation of NDFD Weather Forecasts for Wildland Fire Behavior Prediction. Weather & Forecasting, 33(1), 301–315. https://doi-org.proxygsu-nga1.galileo.usg.edu/10.1175/WAF-D-17-0121.1

Prieto Herráez, D., Asensio Sevilla, M. I., Ferragut Canals, L., Cascón Barbero, J. M., & Morillo Rodríguez, A. (2017). A GIS-based fire spread simulator integrating a simplified physical wildland fire model and a wind field model. International Journal of Geographical Information Science, 31(11), 2142. https://doi.org/10.1080/13658816.2017.1334889

Xie, Y., & Peng, M. (2019). Forest fire forecasting using ensemble learning approaches. Neural Computing & Applications, 31(9), 4541–4550. https://doi.org/10.1007/s00521-018-3515-0

https://medium.com/better-programming/how-to-write-conwells-game-of-life-in-python-c6eca19c4676

Appendix

Code

```
# adapted (for generating and interpreting randomized raster data) from Game of Life code written by Martin A. Aaberge
from random import randint
class Cell:
    def __init__(chance): # Class holding initial status of cells (0 to 99 chance of ignition). Ability to set and fetch new statuses with functions "set" and "get"
        chance._status = randint(0,99)
    def set_newvalue(chance): #sets the cell status to newvalue#
        if chance._status < 25:
            chance._status = chance._status
        elif chance._status in range(25,40):
            chance._status = chance._status + 1
        elif chance._status in range(40,70):
            chance._status = chance._status + 5
        elif chance._status in range(70,99):
            chance._status = chance._status + 20
        else:
            chance._status = 100
    def set_oldvalue(chance):
        chance._status = chance._status
    def set_Fire(chance): #sets the cell status to Fire#
        chance._status = 100
    def is_Fire(chance): #checks if the cell is on fire returns True if it is Fire, False if not#
        if chance._status > 99:
            return True
        return False
    def get_print_character(chance): #returns character to print on the board#
        if chance.is_Fire():
            return 'X'
        return ' '

class Board: #(raster)#
    def __init__(chance , rows , columns): #constructor populates the grid with cells.#
        chance._rows = rows
        chance._columns = columns
        chance._grid = [[Cell() for column_cells in range(chance._columns)] for row_cells in range(chance._rows)]
```

```
        chance._generate_board()
    def draw_board(chance): #draws the actual board in the terminal#
        print('\n'*10)
        print('printing board')
        for row in chance._grid:
            for column in row:
                print (column.get_print_character(),end='')
            print () #creates a new line pr. row#
    def _generate_board(chance): #sets the random state of all cells#
        for row in chance._grid:
            for column in row: #there is a 0.5% chance the cells spawn as fire.#
                firestart_number = randint(0,199)
                if firestart_number == 1:
                    column.set_Fire()
    def update_board(chance): # updates the board based on the check of each cell previous
generation and cells list for non-burning cells to burn and cells to keep burning#
        goes_Fire = [] #empty lists to put cells into#
        gets_newvalue = []
        keeps_oldvalue = []
        for row in range(len(chance._grid)):
            for column in range(len(chance._grid[row])): #check neighbor pr. square#
                check_neighbor = chance.check_neighbor(row , column)
                burning_neighbors_count = []
                for neighbor_cell in check_neighbor: #check status for neighbor_cell#
                    if neighbor_cell.is_Fire():
                        burning_neighbors_count.append(neighbor_cell)
                cell_object = chance._grid[row][column]
                status_main_cell = cell_object.is_Fire()
                if status_main_cell == True: # If the cell is burning keeps it burnt, otherwise checks
the neighbor status#
                    goes_Fire.append(cell_object)
                else:
                    if len(burning_neighbors_count) < 1:
                        keeps_oldvalue.append(cell_object)
                    elif len(burning_neighbors_count) >= 1:
                        gets_newvalue.append(cell_object)
        for cell_items in goes_Fire: #set cell statuses#
            cell_items.set_Fire()
        for cell_items in gets_newvalue:
            cell_items.set_newvalue()
```

```
        for cell_items in keeps_oldvalue:
            cell_items.set_oldvalue()
    def check_neighbor(chance, check_row , check_column): #method that checks neighbors of all cells, lists valid neighbors so the update method can set the new status. min max is depth of search#
        search_min = -1
        search_max = 2
        neighbor_list = [] #an empty list to append neighbors into#
        for row in range(search_min,search_max):
            for column in range(search_min,search_max):
                neighbor_row = check_row + row
                neighbor_column = check_column + column
                valid_neighbor = True
                if (neighbor_row) == check_row and (neighbor_column) == check_column:
                    valid_neighbor = False
                if (neighbor_row) < 0 or (neighbor_row) >= chance._rows:
                    valid_neighbor = False
                if (neighbor_column) < 0 or (neighbor_column) >= chance._columns:
                    valid_neighbor = False
                if valid_neighbor:
                    neighbor_list.append(chance._grid[neighbor_row][neighbor_column])
        return neighbor_list

def main():
    user_rows=int(input('how many rows? (20 recommended) '))
    user_columns=int(input('how many columns? (70 recommended) '))
    fire_model_board=Board(user_rows,user_columns)
    #fire_model_board = Board(20,70) #remove three lines above and the first "#" in this line for hard coded raster (board) sized to fit the default size of the command console#
    fire_model_board.draw_board() #runs the first iteration of the board (or raster)#
    user_action = ''
    while user_action != 'q':
        user_action = input('Press enter to add generation or q to quit:')
        if user_action == '':
            fire_model_board.update_board()
            fire_model_board.draw_board()
main()
```

Permalinks

1_http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=edselp&AN=S136481522030952X&site=eds-live&scope=site&custid=ns235470

2_http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=aqh&AN=124895996&site=eds-live&scope=site&custid=ns235470

3_http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=tsh&AN=139006731&site=eds-live&scope=site&custid=ns235470

4_http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=a9h&AN=133389037&site=eds-live&scope=site&custid=ns235470

5_http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=cph&AN=138884707&site=eds-live&scope=site&custid=ns235470

6_http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,shib&db=edselp&AN=S0168192319303612&site=eds-live&scope=site&custid=ns235470

7_ http://search.ebscohost.com.proxygsu-nga1.galileo.usg.edu/login.aspx?direct=true&AuthType=ip,shib&db=edselp&AN=S1364815218300161&site=eds-live&scope=site&custid=ns235470

8_ http://search.ebscohost.com.proxygsu-nga1.galileo.usg.edu/login.aspx?direct=true&AuthType=ip,shib&db=eih&AN=128402279&site=eds-live&scope=site&custid=ns235470

9_ http://search.ebscohost.com.proxygsu-nga1.galileo.usg.edu/login.aspx?direct=true&AuthType=ip,shib&db=edselp&AN=S1364815218304419&site=eds-live&scope=site&custid=ns235470

10_ http://search.ebscohost.com.proxygsu-nga1.galileo.usg.edu/login.aspx?direct=true&AuthType=ip,shib&db=edselp&AN=S1364815219307388&site=eds-live&scope=site&custid=ns235470

11_https://medium.com/better-programming/how-to-write-conwells-game-of-life-in-python-c6eca19c4676