Fire Simulation Model

I will first go over the current framework I have set up for the model in detail. I will then give an overview of how I plan on using this framework to create the simulation model. The first section should be consulted for the exact implementation I will be using, whereas the second section will explain what this implementation will actually be doing.

The first section of my code is a set of editable variables that act as parameters for the simulation model. These values will only be read from the code (it will never change their values) in order to determine many properties of the simulation.

Currently, the available parameters for the user to edit are:

* The number of simulation runs and the length of each run.
* The FFMC value that affects the mean time between fires.
* The ISI value for determining the fires’ rates of spread.
* Parameters for deciding the latitude and longitude coverage of the forest, and the number of stands it is divided into.
* The number of bases used in the simulation as well as their location and the number of air tankers and bird dogs they each have available to dispatch.
* The number and locations of “random points” in the forest to keep track of during the simulation

The next section of the code is for additional read-only variables that the simulation program uses, however these ones should not be edited by the user.

Next, the code outlines the different classes (objects) that will be used in the simulation. They are as follows:

* Fire. This class includes attributes for many details of the fire such as the time it started, was detected (if it was detected), reported, the time the airtanker(s) arrived at it, and if applicable, the time it was controlled. The slope and fuel type of the forest grid with which it started in, as well as its starting location. The fires size and radius at the current simulation time, and at other important ones (when its detected, reported, when the airtanker(s) arrive). Finally it has a method for updating its current size and radius based on the current simulation time, and a method for printing its attributes
* Statistics. This class keeps track of important averages for each run of the simulation so that they can be used to analyze the simulation results when it finishes. It is currently set to keep track of (for each run):
* Average maximum size of each fire
* Average size of the fires at detection (those not detected are ignored)
* Average size of the fires when reported (those not detected are ignored)
* Average time since the previous fire for a new one to start
* Average time for the fires to be detected (those not detected were ignored)
* Average time for the fires to be reported
* Average percent controlled
* Average time for the fires to be controlled (those not controlled were ignored)
* Number of fires
* Average detection percent
* Average distance travelled by all the airtankers (not each individually)
* Average time travelling by all the airtankers (not each individually)
* Average time spent fighting fires by all the airtankers (not each individually)
* Airtanker. This class keeps track of information for each airtanker running in the simulation. It is also used as a resource by simpy so that the airtankers can be requested by fires, and put into a queue if none are available. It keeps track of its cruising, fight and circling airspeeds (all assumed to be constant), its location in the forest, the distance it has travelled and the time it has travelled (will likely be reset when it returns to a base) and the total distance it has travelled, and the total distance it has travelled, waited in queue and fought fires for. Bird dogs will also likely be represented by this class.
* Base. This class holds the base’s location, as well as a list of all the airtankers and bird dogs that originated from it.
* Forest Stand. The forest is a matrix of these forest stand objects. Their location is determined by their position in this matrix. Each forest stand only keeps track of the slope and fuel type in that location of the forest.
* Point. This class represents the points placed in the forest to be kept track of. It holds attributes for the point’s location in the forest, the total number of times it has been burned, and if it has already been burned on a particular simulation run.

The next section of the program lists all of the functions that will by the if \_\_name\_\_ == “\_\_main\_\_” block of the code. They are all needed to run different aspects of the simulation.

I will quickly go through the functions I have currently:

* Returns the expected number of fires in a day based on the FFMC
* Returns the mean time (mins) between fires 60.0 / (expected fires per day / 8.0)
* Return the Euclidean distance between two locations in the forest
* Return the rate of growth based on the fires fuel type and the user given ISI value
* Return a fuel type to be given to the stand (need to change to some probability dist)
* Return a slope for the forest stand (need to change to some probability dist)
* Return a forest of stands with slopes and fuel types based on the previous 2 functions
* Return the slope and fuel type of the stand at the given location in the forest
* Add a newly created to the list of fires for the run. If it is detected, the time it will be detected, the time it will be reported, its starting location and its slope, fuel type and rate of spread are all determined at the time the fire is created.
* Return a list of point objects with their location attributes set
* Return a list of airtanker objects based on the user-chosen values that will be held by the base that they start at for each run
* Return a list of airtanker objects that represent the bird dogs for each base
* Return a list of the user-specified number of bases used in this simulation, each with their user-specified location and list of airtankers and bird dogs (by the previous two functions) set as attributes on creation.
* Updates the statistics class with all the statistics it keeps track of. Will be used at the end of each simulation run.
* A fire generator function that will be used by simpy to continuously call the create fire function after a random time interval based on an exponential distribution with a mean calculated from a previous function’s returned fires per hour value. It will run for a whole simulation run.

Those are all of the functions I currently have implemented. I will next go over the few functions I will create next in order to allow this simulation model to start functioning. All of the ones labelled processes will be functions used by simpy as process in the simulation environment. These have not yet been implemented. Also, some of these functions seem like they may be a bit large, so I may section off parts of them into other functions that they will instead call.

* Fight Fire process. A function that will be called every time an airtanker arrives at a fire to determine the length of time it spends to try and control it and handles the fighting fire part of the simulation. Will determine if fire is controlled or the airtanker needs to quit based on some rules.
* Dispatch Airtanker process. A function that is responsible for requesting an airtanker and determining, based on some rules, the best airtankers to request, and then calls the previous function when an airtanker is available. Will start off with simple rules. Will be responsible for determining what each air tanker does after controlling a fire.
* A main request process that will call the previous function each time in the simulation when it reaches the next fires reported time.
* Circling process. If the airtanker ever needs to stop fighting the fire and fly in circles, this process will be called by the fight fire process.
* Some process that will be called by the main request process for handling the bird dogs and what they do.
* A main function that will be called by the if \_\_name\_\_ == “main” block that controls all the function calls for a single run of the simulation. It will also update the various relevant statistics.
* The main block that sets up the necessary variables and runs calls the main function the required number of simulation runs.
* A function that will be called by the if \_\_name\_\_ == “\_\_main\_\_” block to print out the simulation results in some specified format (may also incorporate some graphing if needed).

These are all of the functions that I currently plan for this simulation model. However, will creating this model I may feel the need to add or change this list. However, I currently believe that if I implement it in this way the simulation will work well, and it will be relatively straightforward to increase the complexity of the simulation by adding rules to the necessary functions. I don’t believe any further requirements of the simulation model would require a large restructure of the code, just either adding rules or the existing functions, or further adding more functions to handle new requirements. However, if the model’s purpose is significantly altered or different than what I am currently trying to implement, that may be necessary.

Fire Simulation Model Overview

I will now do an overview of the process this whole simulation model is being created to perform, and the purpose it serves.

This simulation model is designed to model fires being created throughout a forest, and the response of airtankers and bird dogs from bases, and their attempt to control these fires. It keeps track of some of the important data during the simulation which can then be examined by the user after the simulation has completed. I will now go through the process the simulation model will take, as well as try to state all of the assumptions being made, and possible shortcomings or areas that could be improved if the model were to be made more complex. \*Note that all of this has not yet been implemented. This is just the current plan for how the model will work when completed

This model first creates a rectangular forest based on a user specified latitude and longitude range. This forest is divided into a user-chosen number of rectangular forest stands. Each stand is generated with a random fuel type and random slope attributed to it. Currently all stands are designated to have a C2 Jack Pine fuel type, and a uniform probability of having a slope from between 0-5 degrees. In the future, more fuel types can be added as well as a probability distribution that each stand is composed of each, as well as changing the slope probability distribution to something more accurate. The slope does not currently affect the rate of spread formula (or anything else), but this could be added (or the FBP system may be incorporated) to give a more accurate representation of the rate of spread.

The model next sets up the user-specified number of bases in their chosen places in the forest. At creation each base is given a list of airtankers and bird dogs that it has available to dispatch. The user must specify the speeds of these aircrafts, as well as the maximum time and distance they can fly without returning to a base. Every airtanker and bird dog can be given different values.

The model then creates a set of points at various locations in the forest based on user specifications. These points will track if they ever get burned or not in each run of the simulation, and their locations as burn-rate will be displayed at the end of the simulation runs.

Now that the model has finished setting up the environment, the simulation runs can start. Based on the user-given FFMC value, an expected number of fires per day is calculated. This value is then divided by 8.0 to find the average number of fire each hour, which is then used to calculate the mean time value that will be used in the exponential distribution that determines the how long it takes to create each fire. Each run of the simulation lasts for a user-chosen amount of time, and fires are continuously created during this time. The simulation model does not currently take into account the time of day for determining the average time between each fire. When the fire is created, it is given a random latitude and longitude in the forest based on a uniform probability. The locations of other fires or bases does not currently affect the locations of new fires in any way (so 2 fires may even start at the same location). Based on its starting location, the fuel type and slope of the fire are determined from the forest stand it was placed in. Even if the fire grows into forest stands of other fuel types, it is assumed to keep the rate of spread properties of the initial forest stand. The slope is not currently used for anything. The fires start as points, and then grow at a constant rate based on their rate of spread. The fire is given an 80% probability of eventually being detected. If it will be detected, then an exponential probability with a mean of 2 hours will be used to determine the time of detection for that fire. An exponential probability with a mean of 10 minutes will then be used to determine the time the fire is reported at. All these values are determined at the time of the fires creation. A request for an airtanker will be sent at the fire’s report time. If an airtanker arrives to fight the fire it will be given some probability of eventually being controlled. It needs to be decided whether or not the fire’s rate of spread will play any role while the fire is being fought by an airtanker.

When a fire is reported an airtanker will be requested to go fight the fire. This simulation model first sets up simple rules for determining which airtanker will go to fight the fire. More complex rules will be added later. This simulation will first assume that a single airtanker from the nearest base to the fire will be requested only. If no airtankers are available from this base, then the fire will wait until one is available, even if airtankers are available from other further bases. After fighting a fire, the airtanker will return to the base it was dispatched from. This will later be changed to allow it to go to other bases or straight to other fires based on certain rules. When an airtanker arrives at a fire to fight, if it is controlled, the length of time it takes is calculated based on some function that estimates a distance to the nearest lake, and some function that uses this with the fires current size. If the fire is not able to be controlled, some function will determine the amount of time that the airtanker spends fighting the fire, and how much of an impact it will have on the fire’s size. The airtankers may also be requested to stop dropping on the fire based on some probability distribution and function for time. Later on better rules for dispatching airtankers will be developed, and multiple airtankers may be chosen to fight the same fire. Also, a maximum flight distance, and maximum flight time without returning to a base for the airtankers will be added that affect what they do. Bird dogs will also be incorporated in this simulation.

This process continues until the user-specified time is reached, at which point the simulation ends (even if all the fires have not yet been controlled). Stats for the day are then recorded, and everything is reverted back to its original state at the start of the day for a new run. The user chooses how many runs the simulation model should perform. After all of these runs, the model will print out various statistics based on the data from the simulation runs performed.