Simulator for multi-stage evaluation of cutting plans in the presence of fire risk (temporary author order)

D.L.W.
Graduate School of Management
University of California Davis
Davis CA 95616, USA
Email: dlwoodruff@ucdavis.edu

C.P.
A.W.
Department of Industrial Engineering
Universidad de Chile
Republica 701, Santiago, Chile
Emails: cpais@ing.uchile.cl
aweintra@dii.uchile.cl

D.L.M.
University of Toronto
Toronto, Canada

April 12, 2017

IMPORTANT: before we do a lot of coding, we need to see if there is a fire simulator we can download. I think there might be. C.P: Actually, D.L.M. has one, I also have it.

NOTE: TBD means "to be done" probably by DLW

PYTHON NOTE: I am not careful when I use the word "list."

Python Note 2: mpi4py, Bokeh and TKinter are python packages (very useful).

Python Note 3: A GUI (Graphical User Interface) should be developed in order to interact with the simulator in next iterations.

1 The Idea

In order to evaluate methods for creating forest harvest plans in the presence of forest fires, fire spread must be simulated. We will begin with some assumptions:

1. Harvest plans are issued and then executed each year.
2. Harvesting takes place before fires.
3. Replanting is soon after harvest.
4. Hourly time resolution is sufficient for modeling the spread of fire.
5. Harvest planning is done for *harvest blocks* and fire spread is modeled using *fire cells* that are much smaller.

Any of these assumptions could be relaxed later. Since the time resolution is the most pervasive and most likely to be changed, we will eschew the use of "hour" and "hourly" and instead refer to *time steps* with the understanding that we mean "fire simulation time steps" and in initial experiments, we mean simply "hours" but we won't use that word.

Assumption 1 implies that the basic time step for computing values and costs is one year.
Assumption 3 is relevant for simulations that run for a large number of years.
Assumption 4 is strong, fire can be modeled within minutes, good as a first approach.

Simulator scheme
In order to represent the problem's dynamic, **Figure 1** shows the different events in the forest. At the beginning of every year we have a *HPeriod* (Harvest Period) where we take harvest decisions - before fire dynamics - and every stand/block is replanted soon after these actions. After this *HPeriod*, *FPeriods* (Fire Periods) start, where all the ignitions and fire spread happen. Important is to note that for the moment we are working without a particular time step size, avoiding confusions about the length of each period.

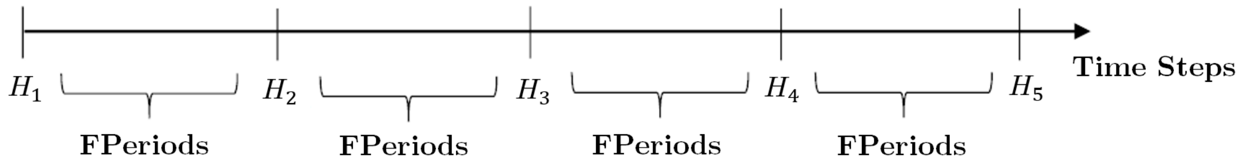


Figure 1: Scheme of fire simulator.

Fire will start at some point - remembering that there is a specific fire season during summer - and then the fire spread model will start to simulate the fire dynamics using the weather and other information inside the simulator (probability of an ignition or fire spread). In **Figure 2**, as an example we can see a forest with 9 stands/blocks where a fire ignites in the first *FPeriod* - after the first *HPeriod* - in cell 4 and then it spreads to different harvest units for two more *FPeriods* (note that the harvest decision was to not touch the forest, since all the forest's cells are available). After these periods, a new *HPeriod* (next year) starts.

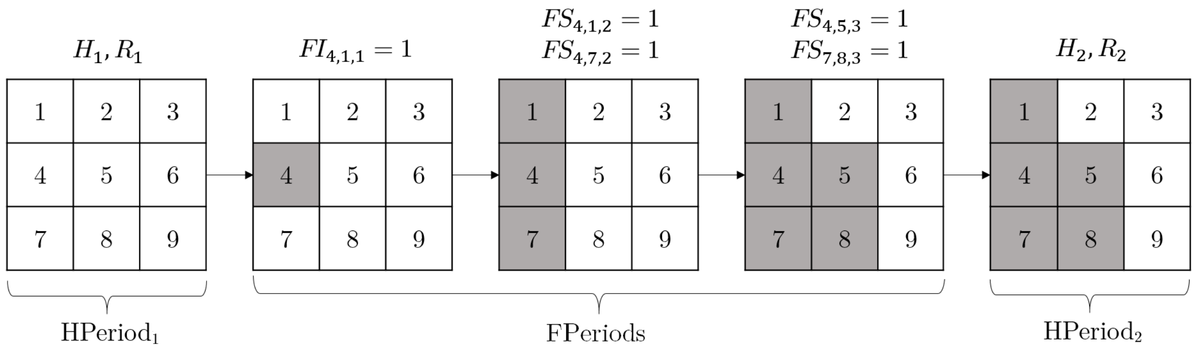


Figure 2: Simulation example.

Where:

- H_{t_h} = Harvest plan for $HPeriod\ t_h$
- R_{t_h} = Reforestation plan during $HPeriod\ t_h$
- $FI_{i,t_f} = 1$ if fire ignites in cell/block i during $FPeriod\ t_f$
- $FS_{j,i,t_f} = 1$ if fire spreads from cell/block j to cell/block i during $FPeriod\ t_f$

Note: *HPeriods* and *FPeriods* do not need to be in the same time scale, we just need them to be consistent. See appendix for details and examples for this idea, applied in a stochastic multi-stage model, including an explanation of *FI* and *FS*.

Then, we can define two main time sets: Harvest Periods (*HPeriods*) indicating the beginning of each year (independent of the time step size) and Fire Dynamic periods (*FPeriods*) where ignitions and fire spreads occur. Important is to note - again - that these sets can use different time scales if they are consistent.

We can define them in a general way:

- Set *Periods* := $\{t \in \mathbb{Z}^+\}$
- Set *HPeriods* := $\{t_h \in Periods : t_h = 1 + \alpha(1+h)$ where $\alpha \in \mathbb{Z}_0^+, h$ is the fire period size$\}$
- Set *FPeriods* := $\{t_f \in \bigcup_{t_h=1}[t_h + 1, t_h + h]\}$

Example: Using 40 periods as the total planning horizon, h = 9 hours and harvesting after every fire period we have:

- Set $Periods := \{1, ..., 40\}$
- Set $HPeriods := \{1, 1 + 1 * 10 = 11, 1 + 2 * 10 = 21, 31\}$
- Set $FPeriods := [2, 10] \bigcup [12, 20] \bigcup [22, 30] \bigcup [32, 40]$

2 Input Data

C.P.: We have to modify this section later, mixing it with section 2.4 Python's Scheme, since a lot of concepts from here are already developed in that section.

2.1 Global data

- Weather, W_t: We are provided weather data for each time step in the year. (Python: a Weather object that has a public function that takes as input a time step number returns a dictionary of weather data (e.g., the keys would be wind speed, wind direction, dew point, temp, etc.)). To get started just have the function return the same weather no matter time step is passed to it.
- Lightning, λ_t: For now, let's just take one lightning parameter which is a global poisson strike rate. It will be assumed this is the rate for lightning that originates from the ground and will start a fire if the fuel and weather conditions permit. (Python: same as weather but simpler: have a Lightening object with a public function, Lambda(t) that, for now, just returns the same value regardless of t)
- Harvested wood values indexed by wood type (TBD: symbol).

2.2 Current harvest block data

(Python: this will come from an object that returns the data in a dictionary given a block ID, it needs the usual initialization methods, but also public methods to allow update of the data in the object. NOTE: this object will just contain current values.)

- Year planted
- Type of harvestable wood
- Volume
- etc.

TBD: symbols

2.3 Current fire cell data

(Python: this will come from an object that returns the data in a dictionary given a block ID, it needs the usual initialization methods, but also public methods to allow update of the data in the object. NOTE: this object will just contain current values.) (more python: there needs to a be fire cell class so there can be a fire cell object for each fire cell and a global fire cell list object).

- Type of fuel
- Terrain (this may, itself be a dictionary in the future)

TBD: symbols

2.4 Python's Scheme

As a first prototype, in this section we present the main elements (Global Objects) in our actual Python script (I will add more here if we need). Both approaches, serial and parallel, use the same objects, they just differ in how objects are created and how they are managed and processed.

Parallel implementation uses MPI (Message Passing Interface) in Python, ready to be used in HPC (High Performance Computers) and in a distributed computing approach in any desktop computer with multiple cores. We exploit parallel programming when: creating objects (initializing the experimental instance), processing them (send and receive message loop) and when writing final results.

Results are processed and passed to a GIS (geographic information system) software in order to track the fire's dynamic in real instances. A series of plots and other statistics are generated and saved as independent files in the working directory. Bokeh package is used here.

1. **Weather Super-Class**
   Weather objects where all parameters like Wind Speed, Wind Direction, Dew Point, Temperature, etc., are computed and updated from an initial state.

   $Obj = Weather(Initial\_Weather)$, where $Initial\_Weather$ is a dictionary.

   (a) **Parameters**
   Mutables
   - $WindSpeed$: Wind speed in [km/hr]. (float)
   - $WindDirection$: Wind Direction in grades (angles). (float)
   - $Temperature$: Temperature in [C^o]. (float)
   - $DPoint$: Dew point in [C^o]. (float)
   - $Rain$: Precipitation in [mm]. (float)

- *Radiation*: Radiation in [J/kg]. (float)

  Example
  $Initial\_Weather = \{$"$Wind\_Speed$" : 24, "$Wind\_Direction$" : 56, "$Temperature$" : 23, "Dew_Point" : 8, "$Precipitation$" : 0.0 , "$Radiation$" : 12$\}$

(b) **Functions & Methods**
  - $set\_Weather(WeatherSet)$:
    Sets a new weather from a dictionary. (void method)
  - $update\_Weather(Period)$:
    Updates the weather based on its current state and forecasts. (void method)
  - $get\_Weather()$:
    Returns the actual weather. (returns dictionary)
  - $print\_info()$:
    Prints weather information (wind, t emperature, etc.). (void method)

Depending on how we are going to implement weather's methods (like update it), we can create a class for each component (not needed for the moment):

- **Class Wind**
  This class extends Weather class. Here, equations to update and define wind will be included.
  (a) Parameters
  (b) Functions & Methods

- **Class Temperature**
  This class extends Weather class. Here, equations to update and define temperature will be included.
  (a) Parameters
  (b) Functions & Methods

- **Class DewPoint**
  This class extends Weather class. Here, equations to update and define dew point will be included.
  (a) Parameters
  (b) Functions & Methods

- **Class Precipitation**
  This class extends Weather class. Here, equations to update and define precipitations will be included.
  (a) Parameters

(b) Functions & Methods

- **Class Radiation**
  This class extends Weather class. Here, equations to update and define radiation will be included.
  (a) Parameters
  (b) Functions & Methods

Current implementation
Each component of the weather is affected by a % depending on a random variable generated from an uniform distribution U(a,b), where a and b are different for each weather component (e.g. $a_{WindSpeed} \neq a_{Temperature}$). Let's call them $\alpha, \beta, \gamma, \delta, \eta$ and θ; associated to wind speed, wind direction, temperature, dew point, rain and radiation, respectively.

Next, every component of the weather is updated based on current values and the random variables (note that these random values can be negative if $a < 0$):

- $Wind\_Speed_t = Wind\_Speed_{t-1}(1 + \alpha)$
- $Wind\_Direction_t = Wind\_Direction_{t-1}(1 + \beta)$
- $Temperature = Temperature_{t-1}(1 + \gamma)$
- $Dew\_point_t = Dew\_Point_{t-1}(1 + \delta)$
- $Rain_t = Rain_{t-1}(1 + \eta)$
- $Radiation_t = Radiation_{t-1}(1 + \theta)$

In a second step, a probability of raining p_r is initialized (by some method) and also, a conditional rain probability q_r is computed like $\mathbb{P}$(Rain in period t/Rain in period $t-1$). Since we are focused in the fire season, the main idea is to keep in mind that it is not so likely to have rain in consecutive periods. Thus, $p_r \geq q_r$.

2. **Lightning Class**
   Fire ignition class, determines the week when fire starts, based on a (non) homogeneous Poisson strike.

   $Obj = Lightning(Period)$.

   (a) **Parameters**
   Inmutables
   - λ_0: Initial fire strike rate [Fires/Fire Season], where Fire Season's length is 12 weeks (3 months). (float)

   Mutables
   - α: Probability factor. (float)

   (b) **Functions & Methods**

- *Lambda_Homogeneous*(*Period*):
  Returns true if a fire starts in a particular week of the fire season using a Poisson distribution. (returns boolean)
- *Lambda_Non_Homogeneous*(*Period*):
  Returns true if a fire starts in a particular week of the fire season, but using a non-homogeneous Poisson distribution. (returns boolean)

  Example (current implementation)
  Let's say that the ignition probability during the fire season is based on a poisson strike $\lambda(t)$ [fires/fire season]. Index t represents the week when fire ignites. Since we know that a fire has to ignite in order to start the simulation, $\lambda(t)$ can be an increasing function on t, having a higher probability of ignition in a week if no ignitions happened before.

  Let:
  $\lambda(t) = \lambda_0(1 + \alpha(t-1)) \qquad (t \geq 1)$
  Then $\Lambda_{(t_a,t_b)} = \int_{t_a}^{t_b} \lambda_t d_t = \frac{\lambda_0}{2}(2(t_b - t_a) + \alpha(t_b^2 - t_a^2 - 2))$

  Since we are dealing with a non-homogeneous Poisson process, we have:
  $N(t) :=$ Poisson counting process, number of fires.
  Probability of having k ignitions until period t: $\mathbb{P}(N(t) = k) = \frac{(e^{-\Lambda_{(0,t)}t})\Lambda_{(0,t)}^k}{k!}$

  Thus, probability of not having an ignition in week 1 is:
  $\mathbb{P}(N(1) = 0) = \frac{(e^{-\Lambda_{(0,1)}})\Lambda_{(0,1)}^0}{0!} = e^{-\Lambda_{(0,t)}}$

  Finally, probability of not having an ignition in week t, if no ignition happened in previous weeks:
  $\mathbb{P}(N(t) = 0/N(t-1) = 0) = \mathbb{P}(N(t) - N(t-1) = 0) = e^{-\Lambda_{(t-1,t)}}$

  Thanks to this expression, ignition probabilities are related to the previous events, instead of the classic "loss of memory" property, present in a homogeneous process.

  Another classic approach is to model the $\lambda(t)$ function with a peak in the middle of the time horizon and then decrease it until the end of the season:

  $$\lambda(t) = \begin{cases} \lambda_0 t, & \text{if } t < \text{Middle of fire season} \\ \lambda_0(\rho_t - t), & \text{otherwise} \end{cases}$$

  Where ρ_t is a parameter that can change through time (or not), allowing us to have different probabilities distributions: symmetric, higher probability in extreme periods, etc.

3. **Forest Class**
   Defines the main parameters of the instance.

$Obj = Forest(ID, Location, Coord, NCells, Area, Vol, Age, Perimeter, FTypes)$

(a) **Parameters**
Inmutables

- ID: Id number. (int)
- $NCells$: Number of cells inside the forest. (int)
- $Area$: Area of the forest [ha]. (float)
- Vol: Volume in [m^3] of cell. (float)
- Age: Age of the forest. (int)
- $Location$: Location of the forest. (string)
- $Coord$: Coordinates of forest's gravity center. (list[float])
- $Perimeter$: Forest perimeter [m]. (float)
- $FTypes$: Types of fuels. (dictionary)
- $CellsDistance$: Distance matrix for cells (gravity centers). (dictionary)

Mutables

- $AvailCells$: List of available (non-burnt) cells. (list[int])
- $BurntCells$: List of burnt cells. (list[int])
- $HarvestCells$: List of harvested cells. (list[int])

(b) **Functions & Methods**

- $set\_AvailCells(Period, AvailCells\_set)$:
  AvailCells list takes the value from the actual AvailCells set (global set). (returns list[int])
- $set\_BurntCells(Period, BurntCells\_set)$:
  BurntCells list takes the value from the actual BurntCells set (global set). (returns list[int])
- $print\_info()$:
  Prints forest information (ID, Location, etc.). (void method)

4. **Cells Class**
Detailed information about each forest's cell, including the send/receive messages functions (fire spread model) and all the math involved when determining a new Fire. This is the most important class of the simulator.
$Obj = Cells(ID, Area, Coord, Age, FType, Terrain, Vol, Perimeter, Status, Adjacents)$

(a) **Parameters**
Inmutables

- ID: Id number. (int)
- $Area$: Area of the forest [ha]. (float)
- Vol: Volume in [m^3] of cell. (float)
- $Coord$: Coordinates of cell [lat,long,alt]. (list[float])
- Age: Age of the forest. (int)

- $FType$: Types of fuel. (int)
- $Terrain$: Type of terrain. (int)
- $Perimeter$: Perimeter of cell. (float)
- $Adjacents$: Adjacent cells divided/classified by cardinal points. (dictionary)

  Examples

  i. FTypeD = $\{1 : \text{“}Normal\text{”}, 2 : \text{“}Burnable\text{”}\}$
  ii. TerrainD = $\{0 : \text{“}Soft\text{”}, 1 : \text{“}Medium\text{”}, 2 : \text{“}Hard\text{”}\}$
  iii. $Adjacents = \{\text{“}N\text{”} :[\cdot], \text{“}S\text{”} :[\cdot], \text{“}E\text{”} :[\cdot], \text{“}W\text{”} :[\cdot]\}$, where $[\cdot]$ are lists that include all the adjacent cells in the four cardinal points, related to the angle between cell's gravity centers (see **Figure 3**).

Mutables

- $Status$: Status of the cell. (int)
- $FICell$: List with Fire ignition parameter. 1 if fire ignites in cell i in period t. (list[int])
- $FSCell$: List with Fire spread parameter. Indicates the adjacent cell's ID if fire spreads from cell $j \in Adjacents_i$ to i in period t. (list[int])
- $HPeriod$: indicates the period where the cell is harvested (if). (int)
- $Firestarts$: indicates the period where the cell is burnt (if burnt). (int)
- $GMsgList$: List with received messages for a cell. (list[lists])

  Examples

  i. Status parameter is associated to a dictionary defined by StatusD = $\{0 : \text{“}Available\text{”}, 1 : \text{“}Burnt\text{”}, 2 : \text{“}Harvested\text{”}\}$.
  ii. $FICell$ =$[0, 0, 1, 0, 0, 0, ...]$, then, fire ignites in period 3 for this cell.
  iii. $FSCell$ =$[[\cdot],[\cdot],[\cdot],[2],[\cdot],...]$, then, fire comes from cell 2 in period 4.
  iv. $GMsgList$ =$[[\cdot],[3,6,7],[1],[\cdot],...]$ indicates that this cell got messages from cells $3, 6$ & 7 in period 2 and a message from cell 1 in period 3.

(b) **Functions & Methods**

- $set\_Status(Status_int)$:
  Changes the status of the cell: 0 available, 1 burnt, 2 harvested. (void method)
- $get\_Status()$:
  Returns cell status. (returns string)
- $send\_msg(Period, Weather, AvailSet)$:
  A cell sends messages to its available (not burnt or harvested) adjacent cells based on weather. (returns list[int])
- $ignition(Period, Lambda, Weather)$:
  Determines if a cell ignites, based on the period, cell characteristics and Poisson distribution. (returns boolean)

- $got\_Burnt(Period, Weather, NMsg)$:
  Based on the number of messages (with at least 1 msg), weather and cell characteristics, determines if the cell gets burnt. (returns boolean)
- $got\_burnt\_from(Period, MsgLists)$:
  Computes FS parameter (see Appendix B). (void method)
- $print\_info()$:
  Prints Cell information (ID, Status, Area, etc.). (void method)

Examples

- $send\_msg(Period, Weather, AvailSet)$
  A cell will send messages to its adjacent cells only if some conditions are satisfied:
  i. Wind Speed $\geq$ WST (wind speed lower bound)
  ii. Temperature $\geq$ TMT (Temperature lower bound)
  iii. Dew Point $\geq$ DPT (dew point lower bound)
  iv. Precipitation $\leq$ RNT (precipitation upper bound)
  v. Radiation $\geq$ RDT (radiation lower bound)

  If these conditions are satisfied, we need to compute a sending message probability, in other words, a fire spread probability. In order to include the time effect, a decreasing probability function on t can be applied, including the $Firestart$ parameter in its definition as:

  $\mathbb{P}(\text{Send a messages in } (t > s)/\text{Cell burnt in } s) = e^{-\frac{t-s-1}{C}}$ where C is a constant that we can modify.

  After computing this probability, we obtain a random number from an U(0,1) distribution and we check if a message is sent. If a message is sent, next step is to evaluate which adjacent cells are going to receive that message, analysing the wind direction. Based on the angle between cells' gravity centers, each cell has an *Adjacents* dictionary where cells are classified in cardinal points (N, S, E and W). These cardinal points are associated to different ranges (see **Figure 3**): $N \in [45^o, 135^o]$, $S \in [225^o, 315^o]$, $E \in [0^o, 45^o] \cup [315^o, 360^o]$ & $W \in [135^o, 225^o]$.

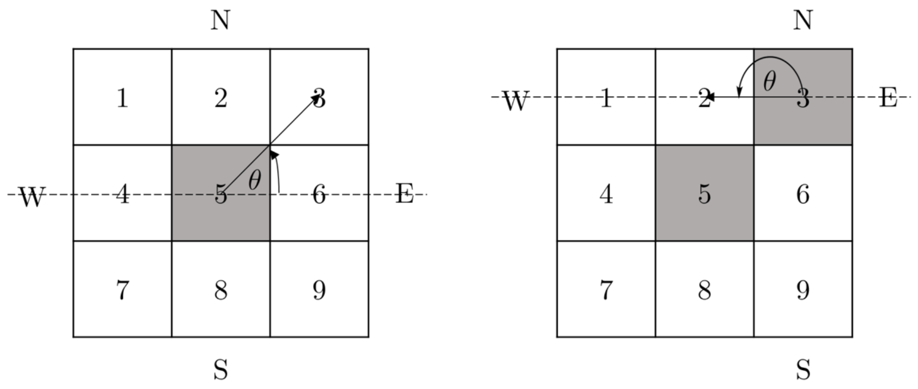


Figure 3: Adjacent cells and angles example. Cell 5 sends a message to cell 3 and then cell 3 sends message to cell 2, based on wind direction.

- $got\_Burnt(Period, Weather, NMsg)$:
  Similar to the previous function, a cell that got a message (or more) from an adjacent cell will get burnt if some weather conditions are satisfied and also, depending on the number of received messages.

  If more than N_{cell} messages are received (a parameter, depending on cell characteristics, type or other elements), fuel type (FType parameter) of the cell is "burnable" and particular weather conditions are satisfied, a cell gets burnt with a very high probability (between 90 to 100%), otherwise, if a cell is not a "burnable type", it still has a chance of getting burnt, based on some probability function.

  The idea behind this scheme is to add stochasticity to the simulation. If we not include these probabilities, every time a cell gets a message, it will have the same behavior/result no matter the number of messages received or fuel type, something that in practice is not so realistic.

5. **Extra Functions**
   Some other functions are needed in order to run the simulator. In particular, we need functions to read all the data (files) from the forest instance, and a special function for parallel implementation.

   - $read\_weather(Path)$
     Reads initial weather data. (returns dictionary)
   - $read\_forest(Path)$
     Reads forest data file. (returns dictionary)
   - $read\_cells(Path)$
     Reads all the information related to cells. (returns dictionary)
   - $jobs\_per\_process(NJobs, Workers)$
     Divides the total amount of jobs (like number of cells to be processed during

the send/receive step in the simulation) in the number of available workers, balancing job queues for each one (MPI implementation). (returns list[int])

6. **Global Parameters**
   Some global parameters are needed in order to develop the simulation, including sets, lists and dictionaries that are updated on every step.

   - *AvailCells_set*: Set that contains all the available cells of the forest. (set)
   - *BurntCells_set*: Set of all the burnt cells of the forest. (set)
   - *HarvestedCells_set*: Set with all the harvested cells of the forest. (set)
   - *GlobalPeriod*: Current stage (week) of the simulation. (int)
   - *FirePeriods*[*Seasons*]: Current fire period of the season. (int)
   - *Seasons*: Current fire season in simulation. (int)
   - *Max_Seasons*: Number of fire seasons to simulate. (int)
   - *Global_Message_List*: Message list, with all the messages senders and receivers (cells) per period. (list[int])

3 Dynamics

C.P.: Same as Input Data main section. We have to modify this section later, mixing it with section 2.4 Python's Scheme, subsection 3.2.2 (Overall view) and section 3.3 (Simulator steps in Python), since a lot of concepts from here are already developed in those sections.

3.1 Fire start

NOTE: this is a slight generalization of AW's idea to "find the week" when it starts, but is the same basic idea.

Loop until a fire starts: Use a competing Poisson model to find the cell and time for the first (or "next") lightning strike; look at the current data for the cell to see if a fire starts. If so start a fire in that cell. In the code: the cell *receives a lightning strike message* and proceeds as described in the fire spread section.

3.2 Fire spread

Once a fire starts each cell does processing for each time step, but only if it gets a message. During the processing, the cell looks at messages sent from the last time step and calculates what happens to it and what messages to send to its neighbors.

An implementation idea: Messages should go in a global message list (that knows what time it is) where each entry in the list gives the sender (not really needed, but good for debugging), receiver, and message.

Example: when the simulator figures out that a fire will start it initializes the message list object with that time step and a message for the fire cell that it has a lightning strike.

Serial version (very close to the parallel version): Fire spread is simulated by giving all the messages in the list (along with the time step number) to the cells that are receiving the messages. After the message is given to the cell, the cell is asked if it has any messages to add to the message list for the next time step. This continues until the next time step is empty (no cell had a message for its neighbors) or a time limit is hit.

NOTE: when I say “list” here, I mean an object that manages a list (which might not be a Python list, but might be).

3.2.1 messages to cell neighbors

Computing what messages to send to neighors is obviously where the whole game is played! We will need help from DLM to do this correctly.

To test: intialze the cells with fuel types “burnable” “not burnable” somehow (not totally random). During fire spread if a cell gets a “heat to you” message and it is burnable, then it burns. If it burns: change it state to burnt. Look at the global wind speed and direction data. If the wind speed is above a threshold, send “heat to you message” to the downwind neighbors.

First enhancement: Have the “heat to you” message include a fire start time and pass heat downwind with a probability that decays as the fire gets older (**Done**).

Second Enhancement: Tests with “medium” type, depending on the number of messages arriving at the same time from different neighbor, e.g. cell 6 can receive a message from cells 5 and 7, then a higher fire probability is computed (**Done**).

Third Enhancement: Probability also depends on a T “decay” factor (**Done**).

3.2.2 Overall view

We need to take into account that there is a fire season, so we do not need to run the simulator for the whole year (saving a lot of time). That info should be included in Lambda(t) (**Done**). Then, using the “fire starts” scheme - loop until a fire starts - , we can identify the date at the fire starts (week also) and proceeds with the fire spread model, using an appropriate time step.

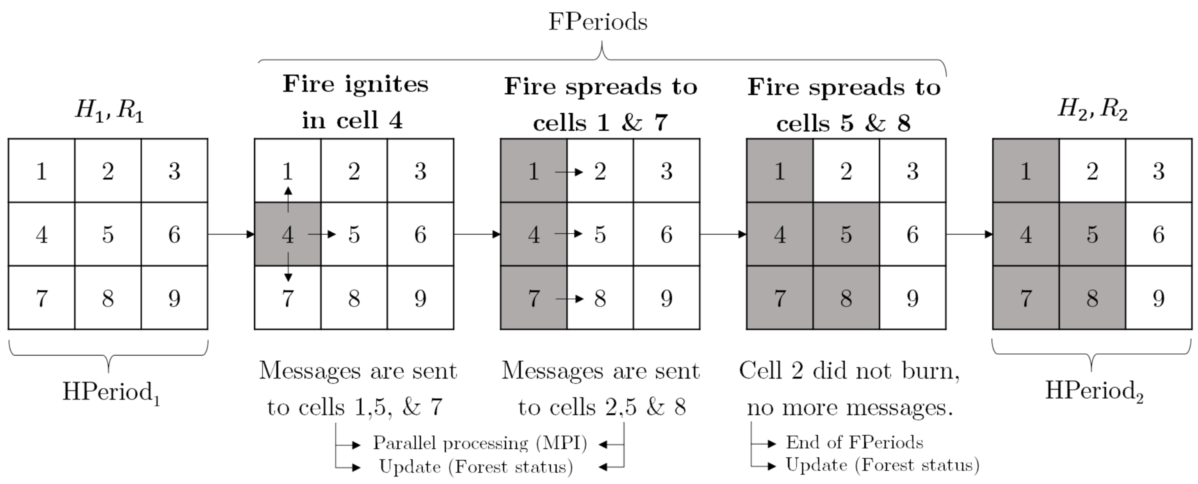

FPeriods
H_1, R_1
Fire ignites in cell 4
Fire spreads to cells 1 & 7
Fire spreads to cells 5 & 8
H_2, R_2
$HPeriod_1$
Messages are sent to cells 1,5, & 7
Messages are sent to cells 2,5 & 8
Cell 2 did not burn, no more messages.
$HPeriod_2$
Parallel processing (MPI)
Update (Forest status)
End of FPeriods
Update (Forest status)

Figure 4: Simulator implementation example.

In **Figure 4** we can see an example of the fire dynamics scheme. Fire ignites in cell 4 during the first *FPeriod* - first message sent to that cell - and this stand starts to send "messages" to some neighbour cells (or to all of them, depending on weather conditions). Cell 4 sends a message to cells 1, 5 and 7. These messages are received by those cells and processed by an independent core/thread in parallel by the simulator - using MPI - in order to check if they are going to be burnt or not. In the next *FPeriod*, cell 1 and 7 got burnt because of the weather and their own conditions while cell 5 not. A global update of the forest status is made (we can use a GIS or a package like Bokeh in Python to represent it in a nice way, plotting the information in a png/jpeg file per period and then mix them to make a "video" of the fire).

In the second FPeriod cells 1,4 and 7 are sending messages to cells 2, 5 and 8 - as a first approach, then we can add the possibility that more than one cell sends a message to the same neighbor, incrementing the probability of getting burnt for that cell - , then parallel processing starts again.

FPeriods end when no more messages are sent by the cells. Forest's status is updated and a new HPeriod starts. Repeat.

3.3 Simulator steps in Python

C.P.: Formal algorithm included in this version
Three main elements will be developed and will be linked from this work. The first element consists of a fire simulator (actual work), allowing us to simulate a general fire's dynamics with many of its complexities in a fast and efficient way (parallel implementation), enabling stochastic evaluation of a harvest planning method or policy. The simulator could also be used to inform the creation of scenarios to be used for stochastic programming.

Both, serial and parallel approaches have been developed and implemented.

Second, a heuristic for yearly harvesting planning will be applied (Appendix A, **C.P: I'm doing final tests**) within the fire simulator, determining which cells have to be harvested in certain periods in order to maximize some benefit function. Third, a multi-stage stochastic MIP model will be developed (Appendix B, **done**) and solved, using Extended formulation and decomposition methods (like Progressive hedging).

Finally, a comparison between different approaches will be done.

1. **Fire Simulation**
   All Python code can be executed from command line or from a GUI (work in progress).

   The simulation code is divided in 5 main steps:

   - **Step 0: Initializing instances, classes, global parameters, sets, etc.**
     Instance data is read from external sources (or can be nested in the code, but not the best option), initializing global parameters and generating all the objects needed for the simulation.

     In parallel implementation, objects are created in parallel using the resources available. Then, if we have N parallel processes, objects generation process will be divided into N job arrays.

   - **Step 1: Lightning/Ignition loop to find the first week with fire**
     Based on the scheme shown in section 2.4, simulation starts when a week is selected as a fire "week". Once this week is selected, a cell, based on its characteristics, status and probabilities, ignites and the fire spread model starts.

   - **Step 2: Burning cells send messages (or not) to their adjacent cells**
     Based on the weather, their status, characteristics and a probability distribution, burning cells can send messages to their adjacent cells. If no messages are sent, end of the simulation. Otherwise, go to Step 3.

   - **Step 3: Messages processing**
     Based on their status, characteristics and a probability distribution function, cells that got messages can get burnt or not. Forest's cells status is updated. Go back to Step 2.

     Critical in parallel implementation where cells will be processed in parallel processes in order to take advantage of the problem structure, saving great amount of time. Cells processing will be divided in a balanced way into the available resources.

   - **Step 4: Results**
     Statistics, animations and plots are generated and recorded in separated files.

A connection with a GIS allows a visual representation of the fire dynamic for real instances.

In the case of plots and animations (gif files representing the fire's dynamics), colors represent the status of the cell: green for available cells, brown for harvested cells, orange for burnt cells and different reds for burning cells. Time decay factor is represented by this colors according to the fire's intensity, thus, when a cell has just got burnt, it has the most intense red (alongside with the highest probability of spreading fire to its adjacent cells in the wind's direction) and every period its color will tend to an orange one. See **section 4.2** for more details.

2. **Heuristic**
   As a second step, a heuristic will be implemented in order to take yearly harvest decisions (at the beginning of each fire season, before fire) **C.P: Added in Appendix A**.

3. **Stochastic Model**
   Finally, a stochastic multi-stage mixed integer problem will be implemented (Appendix B, **done**) and tested for different number of scenarios (generated from the simulator, in progress) and contrasted with the heuristic in terms of performance (solving time) and quality of the obtained solution.

4 Tests & Results

C.P.: I will add them in next versions of this doc, with more plots and statistics.
C.P: I have some forest data (from Chile and Portugal, thousands of stands). I will create small forests (9 and 16 stands) for the first tests and I will start to develop a prototype version of our python script. As a future step, we can test Ignacio's paper data.

4.1 Test Instances

As a first approach, we create small instances with 9, 16, 25, 100 and 10000 cells. Computer generated forests' data (**see Appendix C for details**) is used as input for the simulator. Each instance has a square shape (NxN) and its characteristics are summarized in **Table 1**. When simulation starts, forests have all their cells available (no previous fires or harvest).

1. **3 x 3 cells**
   This instance consists in a forest with 9 cells distributed in a square shape. As a first experiment, all cells present similar characteristics in terms of Age, Area, Productivity and initial status (available). 3 different fuel types ("Normal", "Medium" and

Algorithm 1 Simulator Pseudo-code

```
procedure Fire Simulator($Forest\_Data, Weather, Demand, Periods, TotalRuns$)
    Step 0: Initializing parallel environment
        Determine number of parallel processes
    Step 1: Initializing Instances, classes, global parameters, sets, etc.
        Read forest's data
        if parallel process == True:
            Distribute data among processes
            Initialize Objects and Broadcast them
        else:
            Serial Initialization
    Step 2: Lightning/Ignition loop in order to find the first week with fire
        NoIgnition = False
        while $Fire\_Period < FirePeriods$:
            if strike(Fire_Period) == True:
                $Fire\_start = Fire\_Period$
                $Cells.Status = Cells.Ignition(Fire\_start)$
                Update $BurningCells\_Set(Fire\_start)$
                break, go to Step 3
            else:
                NoIgnition = True
                Next Year, Step 2
            $Fire\_Period + = 1$
    Step 3: Sending messages
        EmptyMsgList = True
        while $BurningCells\_Set(Fire\_Period)$ is not empty:
            for $c \in BurningCells\_Set(Fire\_period)$:
                if $Cells[c].sendMsg$ == True:
                    MsgList = MsgList $\cup$ $Cells[c].msg$
                    EmptyMsgList = False
            $UpdateCells.Status$
            $Fire\_Period + = 1$
            if EmptyMsgList == False:
                go to Step 4
    Step 4: Receiving messages
        for c in MsgList:
            if $Cells[c].gotBurnt$ == True:
                $Cells[c].Status$ = "Burning"
                $BurningCells\_Set(Fire\_Period) \cup \{Cells[c].ID\}$
            go to Step 3
    Step 5: Results and Outputs generation
        Generate Excel and txt files
        Output plots and animation (optional)
end procedure
```

"Burnable") and 3 types of terrain ("Soft", "Normal" and "Hard") are distributed along the forest.

Initial Weather conditions are generated based on historical data, taking into account the fire season characteristics.

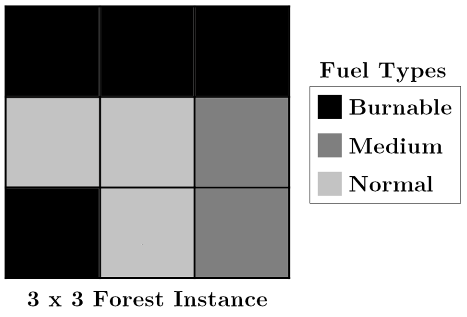


Figure 5: 3×3 instance scheme. Different Fuel Types are included within the forest, following the color scheme on the right of the plot.

2. **4 x 4 cells**
   Similar to the previous instance, 16 cells with 3 different fuel types and terrains are represented in a square shape. The same weather conditions are used.

3. **5 x 5 cells**
   The main characteristic of this instance consists in the limited amount of "Burnable" (Fuel type) cells, covering less than 25% of the entire forest. Because of this condition, ignition and spread to/from probabilities are very low.

4. **10 x 10 cells**
   In contrast to the previous instance, more than 90% of this forest is "Burnable". Thus, larger fires have a very high probability of occurrence, transforming this instance into a critical study case.

5. **100 x 100 cells**
   The largest test instance includes 10,000 cells. In this case, a higher productivity factor is applied for testing purposes. Again, we include 3 types of fuel and terrain. Due to its size, simulator's performance, both in parallel and serial approach will be tested.

Instance	**Total Cells**	**Avg Age**	**Fuel Types**	**Avg Area**	**Avg Productivity**
3 x 3	9	10	3	20	12
4 x 4	16	12.2	3	16.2	12
5 x 5	25	12	3	15.9	12
10 x 10	100	14.5	3	14.2	12
100 x 100	196	14.3	3	19.4	24

Table 1: Test instances description summary

4.2 Real Instances

Real forests' data is tested in the simulator. Previous processing work must be done in order to translate the original data into the simulator's format, ready for fire simulations.

A summarized description of each instance is in **Table 2**.

- **Instance 1: Chile**
  Chilean Instance description...
- **Instance 2: Portugal?**
  Portuguese Instance description...
- **Instance 3: Canada?**
  Canadian Instance description...

Instance	Total Cells	Avg Age	Fuel Types	Avg Area	Avg Productivity
Chile	256	18	3	55.2	
Portugal 1	354	13.4	3	33.2	
Portugal 2	1000	15.8	3	38.4	
Canada	-	-	-	-	-

Table 2: Real instances description summary

4.3 Results

Examples of four fire's periods simulations are presented in this section. A total number of 10,000 simulations are run for each instance, allowing us to perform statistical tests and obtain rigorous and representative results.

All instances start with the same initial conditions (weather and forest status, all cells available). Statistics and plots are generated and recorded.

Results obtained from the all test and real instances are summarized in **Tables 5 and 6**.

1. **Test Instances**
   - **3 x 3 cells**
     After each replication, forest's final status is recorded. At the end of the whole simulation process, available cells and burnt cells averages values are reported. Alongside with this, the empirical fire frequency for each cell is computed, giving us an approximation of their fire probabilities. This information will be

used to set up the value of the weights needed to compute the cells' value function inside the harvest heuristic (**Appendix A**).

Based on the results (**Table 3**), in average a 57.78% of the forest is available by the end of the planning horizon (4 years) and a 42.22% got burnt. Cells 1,2,3 and 6 have the highest fire relative frequency (Number of fires in cell i/Total number of simulations) with a value of 97%, followed by cell 7 with 90%. The cells with the lowest probability of getting burnt are cells 4 and 9 (3% and 6%).

3 x 3 Forest simulation results		
Final Status	Available Cells (AVG)	3.8
	Burnt Cells (AVG)	5.2
Fire relative frequency [%]	Cell 1	0.95
	Cell 2	0.97
	Cell 3	0.97
	Cell 4	0.03
	Cell 5	0.23
	Cell 6	0.97
	Cell 7	0.90
	Cell 8	0.17
	Cell 9	0.06

Table 3: 3 x 3 instance results.

Then, the overall fire risk for the whole forest when no harvesting (or immediate actions) decisions are made can be quantified in terms of its average final status after a series of simulations, identifying those cells with higher fire risk and those who can act as potential firewalls. These results are used to construct the harvest heuristic that we want to apply in a second state of the project (**Appendix A**).

In a third step of the project, all the information obtained from the simulation will be used to represent a series of fire scenarios (with their own probability of occurrence), that will act as input data for a multi-stage stochastic problem (**Appendix B**).

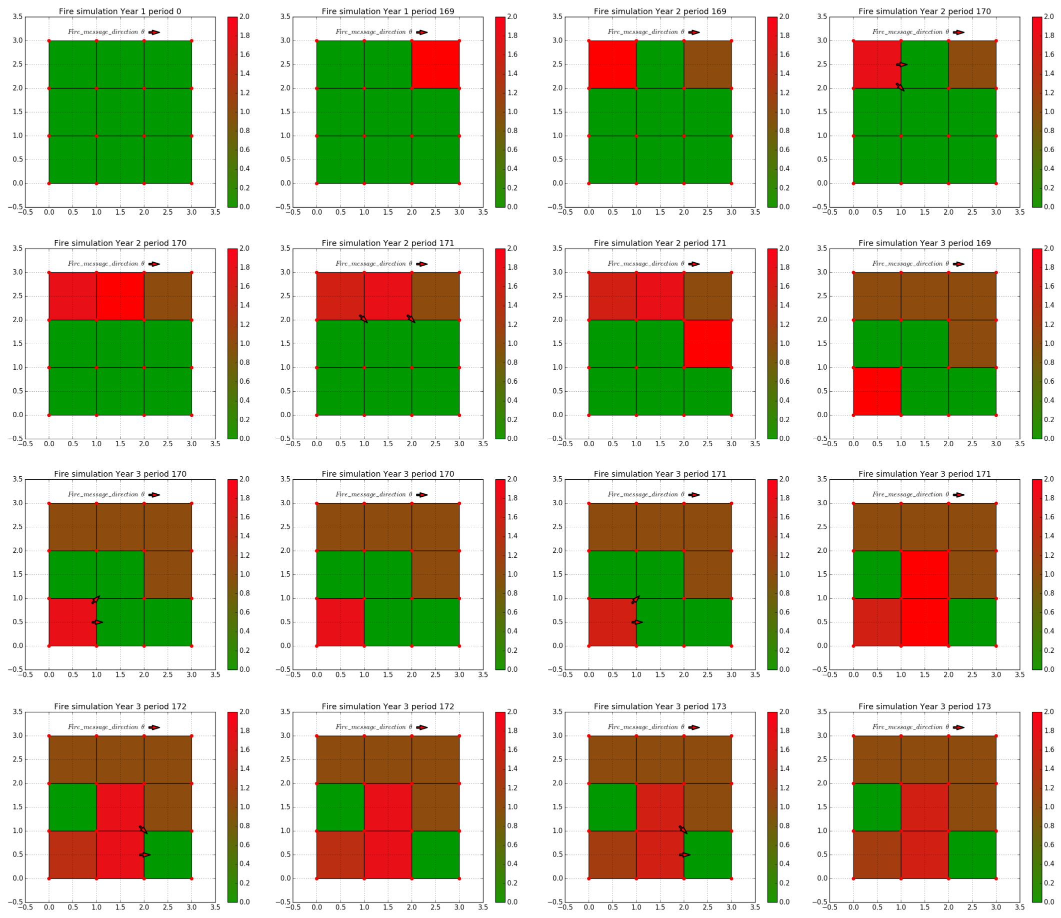


Figure 6: Example of simulation's output in 3 x 3 test instance.

In **Figure 6**, a 4 seasons simulation's output is shown. At the beginning of the simulation (year 1, period 0) all cells are available (no previous fire or harvest). Then Cell 3 ignites during the first fire season, in period 169 (from a total of $12 \times 7 \times 24$ periods, total hours of the 3 months fire season). Due to weather conditions (wind's direction, speed, temperature, dew point, etc.), no more cells got burnt during this season. Note that at the beginning of the second year, cell 3 is brown since it is already burnt.

Cell 1 ignites in the second year, spreading fire to cell 2 and then, cell 2 spreads it to cell 6. No more cells got burnt in this period. In year 3, cell 7 ignites and spreads fire to cells 5 and 8 (red arrows in the plot indicate fire messages' direction). Different shades of red indicate time and temperature decay factor, where most intense red is associated to the highest "spread to" probability. By the end of the simulation, burnt and available cells are shown (**Figure 7**).

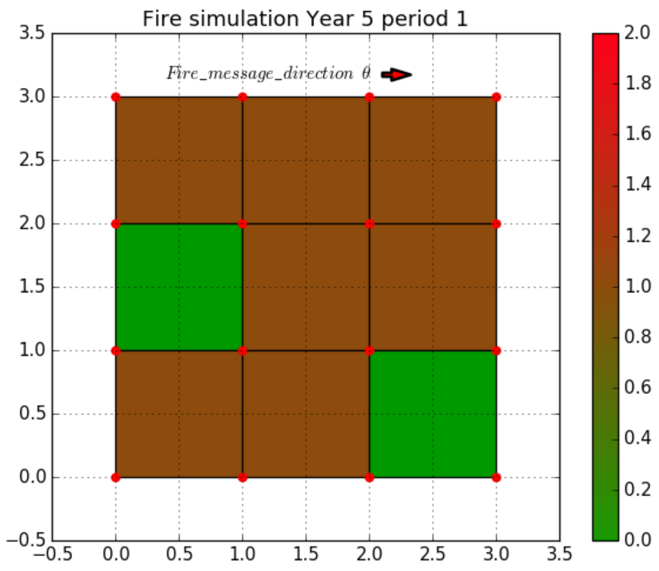


Figure 7: Final state of a simulation in 3 x 3 test instance.

No significant difference in terms of running time is achieved when using a parallel approach since the size of the forest is too small to take real advantage of this scheme.

- **4 x 4 cells**

  Following the same methodology applied in the previous instance, simulation results are recorded and analysed for the 16 cells.

  Based on the results (**Table 4**), in average a 51.1% of the forest is available by the end of the planning horizon (4 years) and a 48.9% got burnt. Cells 3 and 7 have the highest fire relative frequencies (Number of fires in cell i/Total number of simulations) with values of 94.5% and 93.1%, followed by cells 12, 6, 11 and 1 with 91%, 87%, 86% and 85%. The cells with the lowest probability of getting burnt are cells 5,9 and 13 (1%).

  In **Figure 8**, a 4 seasons simulation's output is shown. At the beginning of the simulation (year 1, period 0) all cells are available (no previous fire or harvest). Then Cell 1 ignites during the first fire season, in period 169 (from a total of $12 \times 7 \times 24$ periods, total hours of the 3 months fire season). Due to weather conditions (wind's direction, speed, temperature, dew point, etc.), cells 2,3,4,6,7,8,11 and 12 got burnt (due to fire's dynamics) during the first period.

  No more ignitions occurs in the following periods. By the end of the simulation, 9 cells are burnt (56.25% of the forest).

4 x 4 Forest simulation results		
Final Status	Available Cells (AVG)	8.2
	Burnt Cells (AVG)	7.8
Fire relative frequency [%]	Cell 1	0.85
	Cell 2	0.93
	Cell 3	0.95
	Cell 4	0.25
	Cell 5	0.01
	Cell 6	0.87
	Cell 7	0.93
	Cell 8	0.32
	Cell 9	0.01
	Cell 10	0.02
	Cell 11	0.86
	Cell 12	0.91
	Cell 13	0.01
	Cell 14	0.67
	Cell 15	0.11
	Cell 16	0.17

Table 4: 4 x 4 instance results.

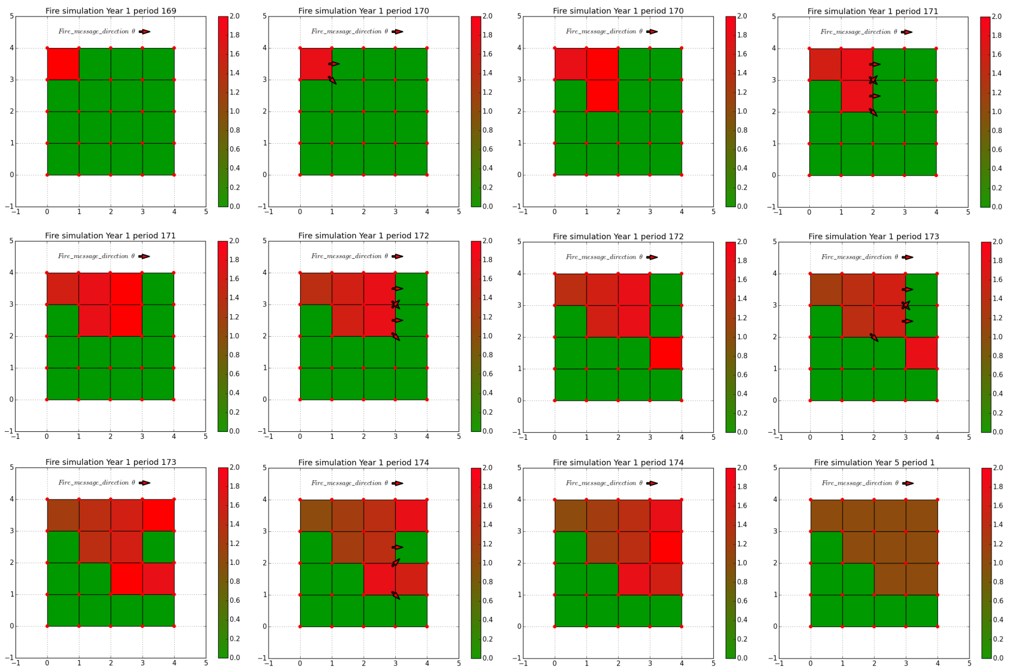


Figure 8: 4 x 4 cells forest's simulation example.

- **5 x 5 cells**
  Following the same methodology applied in the previous instance, simulation

results are recorded and analysed for the 25 cells.

- **10 x 10 cells**
  Following the same methodology applied in the previous instance, simulation results are recorded and analysed for the 100 cells.

- **100 x 100 cells**
  Following the same methodology applied in the previous instance, simulation results are recorded and analysed for the 10,000 cells.

2. **Real Instances**

   - **Instance 1: Chile**

   - **Instance 2: Portugal?**

   - **Instance 3: Canada?**

3. **Summary**
   Insert analisys here...

Instance	Burnt cells	Available cells	Sim Time (serial) [s]	Sim Time (4C parallel) [s]
3 x 3	5.2	3.8		
4 x 4	7.8	8.2		
5 x 5				
10 x 10				
100 x 100				

Table 5: Test instances results for 10,000, in serial and parallel simulation approaches

Insert analisys here...

Instance	Burnt cells	Available cells	Sim Time (serial) [s]	Sim Time (4C parallel) [s]
Chile				
Portugal 1				
Portugal 2				
Canada				

Table 6: Real instances results for 10,000, in serial and parallel simulation approaches

5 Conclusions

-
-
-
-
-

6 Appendix A

C.P: Done.
Harvest Heuristic: Implemented in the simulator.

Once the fire simulator is finished, a harvesting heuristic to determine the yearly forestry planning is developed. The goal is to obtain the maximum "benefit" from the forest, taking into account the possible fire's dynamics (probabilities of ignition and spread, from and to neighbors) inside of it, forest's characteristics (terrain, fuels types, etc.), cells' spatial distribution (interaction with neighbors) and weather effects.

In order to determine the optimal harvesting plan, the heuristic uses a series of parameters and variables from the forest: number of available, harvested and burnt cells per year, expected weather for the next fire season and probabilities information.

The heuristic follows a "rolling horizon" behavior, taking decisions at the beginning of each year based on the available forest's data.

The scheme is as follows: In order to determine which cells are going to be harvested each year, a cell's ranking is developed based on an objective function **(see subsection 6.0.1 and 6.0.2)** generated from forest's data and future fire probabilities that allows us to select which cells will be harvested in a "top to down" procedure, until minimum demand bound (for that year) is achieved.

Thus, cells with better ranking are going to be harvested before. Cells' values will change year by year based on forest's status. Then the most important step is how to quantify cells' value in a way that their economic value is included, but at the same time, information and expectations about fire probabilities (both ignition and spread ones) is also taking into account. We do this based on the logical assumption that a higher probability of fire could lead to larger fires in the forest - leaving less available cells for next years -, with its corresponding monetary and environment losses.

C.P.: I tested it, preliminary results are included

6.0.1 Model

The heuristic's value function is composed by the following elements:

1. **Ignition Factor**
   Cell's ignition probability is weighted by the "Adjacents ratio" that we define as $\frac{|AvailAdjCells_{i,t}|}{MaxAdjacentCells}$ in order to represent the potential fire spread risk of the cell i. If a cell i ignites and has more available adjacent cells than the rest of cells, it has a higher probability of spreading fire to one (or more) of them during the fire season, leading to higher fire risk. This value is divided by the maximum number of adjacent cells a cell i can have (8 cells if we are dealing with a grid shape) to measure the potential effect of an ignition in a particular cell, for the whole forest.

   The factor expression is:

$$\mathbb{P}(ignition)_{i,t} \times \frac{|AvailAdjCells_{i,t}|}{MaxAdjacentsCells} \qquad \forall i \in Cells, \forall t \in HPeriods(years)$$

2. **"Spread to" Factor**
   This component represents the risk of fire when a cell ignites and starts to send fire messages to its neighbours/Adjacent cells, following the weather conditions (wind speed, wind direction, temperature, dew point, radiation, etc.).

   Each cell i has particular characteristics that are used in order to compute its "spread to" probability. Since a cell would spread fire to its adjacent cells that **are in the wind's direction**, we weighted this probability with the number of cells's i available adjacent cells that are in the expected wind direction, divided by the total number of its available adjacent cells, in order to measure the potential impact of having a fire in a cell i **(see Figure 9)**. Then, this expression is multiplied by the "Adjacents ratio" presented in (1).

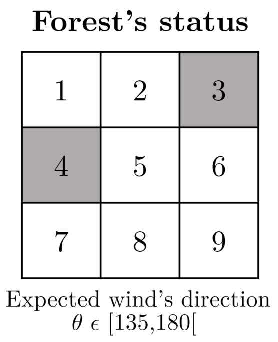


Figure 9: Based on wind's direction and forest's status, cells 3 and 4 are burnt. Using the expected wind direction (North West in this case), the ratio $\frac{|ProWindAvailAdjCells_{i,t}|}{AvailAdjacentsCells_{i,t}}$ will be $\frac{1}{3}$ for cell 2 and $\frac{1}{4}$ for cell 8.

The "Spread to" expression is:

$$\mathbb{P}(Spread\ to)_{i,t} \times \frac{|ProWindAvailAdjCells_{i,t}|}{AvailAdjacentsCells_{i,t}} \times AdjacentsRatio$$
$$\forall i \in Cells, \forall t \in HPeriods(years)$$

3. **"Spread from" Factor**
   Similar to the previous component, this element represents the fire risk when an adjacent cell ignites and starts to send "fire messages" to cell i, based on weather conditions and cells' characteristics.

   Since burning cells would spread fire to adjacent cells that are in the wind's direction **(see Figure 10)**, this probability is weighted by the number of available adjacent cells that are in the **opposite direction**, divided by the total number of available cells. Again, this expression is multiplied by the "Adjacents ratio".

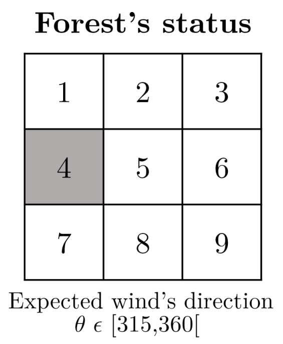


Figure 10: Based on wind's direction and forest's status, cells 5 and 8 can receive fire messages from cells 1 or 7 (possible burning cells). Using the expected wind direction (South East in this case), the ratio $\frac{|AgaintsWindAvailAdjCells_{i,t}|}{AvailAdjacentsCells_{i,t}}$ will be $\frac{2}{7}$ for cell 5 and $\frac{1}{4}$ for cell 8.

The "Spread from" expression is:

$$\mathbb{P}(Spread\ from)_{i,t} \times \frac{|AgaintsWindAvailAdjCells_{i,t}|}{AvailAdjacentsCells_{i,t}} \times AdjacentsRatio$$
$$\forall i \in Cells, \forall t \in HPeriods(years)$$

4. **Adjacents factor**
We include the "Adjacents ratio" as an isolated factor, representing the potential influence/rol of a cell inside the forest when a fire occurs. When a cell has more adjacent cells (available) it has a higher potential fire risk, since a fire can start because of its ignition (and then it can be spread to the rest of the forest) or an adjacent cell can spread fire toward it, generating a larger fire.

Thus, this component allows us to take into account these potential interactions.

The expression is:

$$\frac{|AvailAdjacentCells_{i,t}|}{MaxAdjCells} \qquad \forall i \in Cells, \forall t \in Periods$$

5. **Net Benefit**
Economic benefits from harvesting cells is included as a component of the value function. It is computed as follows:

$$U_{i,t} = Benefit_{i,t} - HarvestingCost_{i,t} \qquad \forall i \in Cells, \forall t \in HPeriods(years)$$

$$Benefit_{i,t} = Price_t \times Cell.Area_{i,t} \times Cell.Productivity_{i,t}$$
$$HarvestingCost_{i,t} = Cell.HC_{i,t} + Cell.HCV_{i,t} \times Cell.Area_{i,t}$$

Where:

- $Price_t$ = Market price for $HPeriod\ t\ \frac{[USD]}{m^3}$.
- $Cell.Area_{i,t}$ = Cell's i available area during $HPeriod\ t$.

- $Cell.Productivity_{i,t}$ = Cell's i productivity during $HPeriod\ t$ $\frac{[m^3]}{[m^2]}$.
- $Cell.HC_{i,t}$ = Cell's i fixed harvesting cost in $HPeriod\ t$ [USD].
- $Cell.HCV_{i,t}$ = Cell's i variable harvesting cost in $HPeriod\ t$ $\frac{[USD]}{[m^2]}$.

6. **Weights** (β)

   In order to compute the value function, all previous components are weighted by a weight's vector. Each component of this vector (β_f, $\forall f \in \{1, ..., 5\}$) represents the relative importance of each factor inside the global cell's value function, where $\sum_{f=1}^{5} \beta_f = 1$.

   How to select those weight is not an easy task, since it depends on the decision maker's aims. For example, the decision maker would want to minimize the number of burnt cells by the end of the planning horizon, despite the total harvesting benefits (assigning more weight to factors 1-4). On the other hand, a decision maker would prefer to assign more weight to monetary elements (component 5), because fire risk could be very low in his forest.

   In this work, we want to find the weight's vector that provides the best results when analysing the trade-off between fire risk and economic benefits within the forest in a way that we seek to maximize forest's expected value.

   A series of tests must be performed in order to determine these optimum values. Details of these tests and the applied approach are in **section 6.0.3**.

Thus, using these five components, the value function for a cell i in $HPeriod\ t$ is computed as follows:

$$VallFunction_{i,t} = \sum_{f=1}^{5} \beta_f \times Factor_f \qquad \forall i \in Cells, \forall t \in HPeriods$$

where $\sum_{f=1}^{5} \beta = 1$

6.0.2 Algorithm

The pseudo-code of the heuristic is as follows:

This Heuristic runs every simulation, for every year of the planning horizon. Statistics (excel and txt files) and plots (png files) are recorded after each simulation. A comparison of the final forest's status when using the heuristic and when no decisions are taken is performed.

Algorithm 2 Heuristic

```
1: procedure HARVEST HEURISTIC($Forest\_Data, Weather, Demand, Period$)
2:     $Year \leftarrow Period$
3:     $(AvailableSet, BurntSet, HarvestedSet) \leftarrow Forest\_Data$
4:     $Weather \leftarrow Weather.getWeather$
5:     $Harvested[Year] \leftarrow 0$
6:     $Benefit[Year] \leftarrow 0$
7:     $CellValue[Year] \leftarrow 0$
8:     for $c \in AvailableCells_Set()$ do
9:         $CellValue[Year] \leftarrow Heuristic.CellVal(c)$
10:    end for
11: While $Harvested[t] < Demand[t]$ and $Val(AvaillableSet) < \epsilon$ :
12:    $[ID, maxCellVal] \leftarrow max(CellValue[Year])$
13:    $AvailableSet \leftarrow AvailableSet \setminus \{ID\}$
14:    $HarvestedSet \leftarrow HarvestedSet \cup \{ID\}$
15:    $Cell[ID].Status \leftarrow$ "Harvested"
16:    $Harvested[Year] += Cells[ID].Volume$
17:    $Benefit[Year] += Benefit(Cells[ID])$
18: end procedure
```

6.0.3 Weights β

How to determine the heuristic objective function's weights is the most important step when developing the heuristic, since a cell will be harvested only if its "value" is on the top of the heuristic's ranking. The decision maker can select them by experimentations, running a series of simulations with different weights vectors in order to find the "optimum one" for his interest (economical, environmental or a mix), but this approach is not useful in practice since it will spend a significant amount of time - alongside with computational resources - and there is no quantitative explanation behind those values.

Thus, the main question - and challenge - is how to select these weights. In order to address this problem, we propose the following methodologies:

Methodology I: Expected βs

1. **Simulation**
   A series of simulations (10,000) are performed for each instance, recording each scenario data - ignitions, fire's dynamics and scenario probability of occurrence - in the format needed to use a Stochastic Model per scenario (**see Appendix B**). This model will take all scenarios' data as input for an optimization procedure.

2. **Stochastic Model per scenarios**
   Scenarios' information is summarized in both ignition and spread parameters (FI and FS), allowing the model to take decisions (harvest decisions) based on the

forest's future fire. Fire's dynamics are represented as constraints inside the optimization model (**see Appendix B**) and extra constraints regarding the final status of the forest, like a the maximum percentage of available/burnt forest by the end of the planning horizon, must be added in order to find an optimal solution.

After solving each scenario, we will have $|S|$ solutions ($\overleftarrow{H_s}$, scenarios' decisions vectors). Each solution contains the optimal harvest decisions for each scenario for the whole planning horizon. It is interesting to note that these decisions can include as many extra constraints as a decision maker would want, adding flexibility to this approach. As an example, a minimum profit and maximum burnt percentage constraints can be added to the model, alongside with other considerations (water runoff, carbon emissions, etc.).

If we want to replicate these solutions when using the heuristic inside the simulator, we must determine the objective function weights ($\overleftarrow{\beta}$) in a way that cells with higher value (per period) are the same cells as the ones that are harvested in the stochastic solution. Thus, in order to find these weights, we use these solutions as inputs inside an auxiliary adjustment model.

3. **Auxiliary Model: Beta's adjustment**
   The auxiliary adjustment model is as follows:

   (a) **Sets**
   - $HPeriods\{t_h\}$
   - $Cells\{i\}$
   - $HFix_{t_h}\{c_{t_h}\}$
   - $Betas\{f\}$

   (b) **Variables**
   - β_f = value of weight f, $\beta_f \in \mathbb{R}_0^+ \ \forall f \in Betas$
   - δ_{t_h} = Auxiliary variables per $HPeriod \ t_h$, $\delta_{t_h} \in \mathbb{R}_0^+ \ \forall t_h \in HPeriods$

   (c) **Parameters**
   - $Factor_{f,i,t_h}$ = Objective function factors for weight f, for cell i, during $HPeriod \ t_h$

   (d) **Formulation**

   $$\max \text{ Difference} = \sum_{t_h \in HPeriods} \delta_{t_h}$$

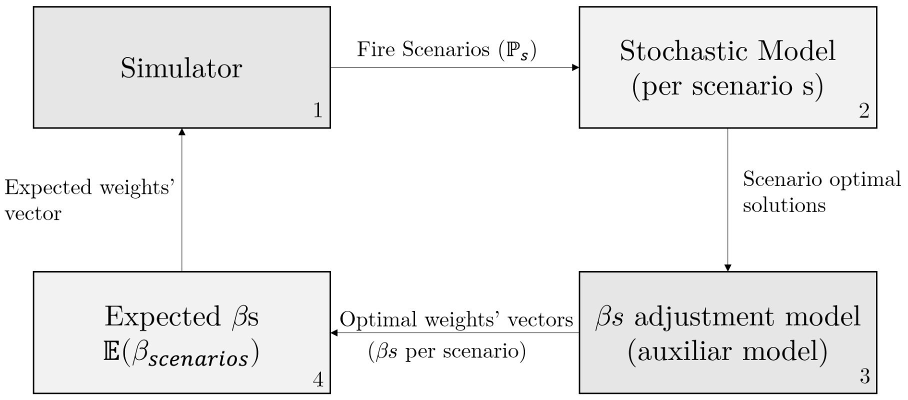
   s.t

   i. $$\sum_{f \in Betas} \beta_f \times Factor_{f,c,t_h} \geq \sum_{f \in Betas} \beta_f \times Factor_{f,j,t} + \delta_{t_h}$$
   $$\forall t_h \in HPeriods, \forall c \in HFix_{t_h}, \forall j \in Cells : j \notin HFix_{t_h}$$

   ii. $$\sum_{f \in Betas} \beta_f = 1$$

The main idea of the model is to maximize the difference between cells' values when evaluating them using the heuristic objective function. Using the solutions obtained from the stochastic model as fixed solutions (represented by $HFix_{t_h}$), it is possible to add a set of constraints that model the relationship within cells' values. Thus, cells that are harvested in the stochastic solution, must have higher values in the heuristic (see constraint (i)) in order to be harvested in particular periods of the planning horizon.

Since the heuristic selects which cells are going to be harvested based on their value and the period's demand, an alternative harvest criterion must be developed in order to add the possibility of harvesting **more** than the current period's demand. This situation can occur when the exact solution harvests more cells in a particular period (usually in early periods) in order to protect a wider forest's area from future fires. Thus, a "value threshold" is added as part of the heuristic, allowing it to harvest cells that have high values (larger than the threshold ϵ) although the period's demand has been already satisfied.



Simulator
1
Fire Scenarios ($\mathbb{P}_s$)
Stochastic Model (per scenario s)
2
Scenario optimal solutions
βs adjustment model (auxiliar model)
3
Optimal weights' vectors (βs per scenario)
Expected βs $\mathbb{E}(\beta_{scenarios})$
4
Expected weights' vector

Figure 11: βs' adjustment methodology steps.

By the end of this step, we will have $|S|$ weight's vectors, where S is the set of scenarios. Each weight's vector consists in the βs needed to obtain the same harvesting plan as the exact model when using the heuristic within the simulator. In other words, it is the "translation" between a full information model (stochastic model) and a "wait and see" heuristic.

4. **Expected β's**
Once we have all the optimal weights vectors - for each scenario - we can compute the optimal expected weight vector (using scenario probabilities) for the heuristic. These weights should reach the best expected result when applying the heuristic in a random scenario since they take into account a wide range of fire possibilities (possible scenarios). Thus, thanks to this adjustment, the heuristic should improve its performance, both in economical and forest conservation terms (depending on the constraints used in the stochastic model).

5. **Heuristic implementation**
   Finally, the expected weights will be implemented inside the heuristic. Important is to remember the threshold (ϵ) parameter that allows the heuristic to harvest (or not) more cells than the ones needed to satisfy the current demand, in order to protect some sections of the forest from future fires.

Methodology II: Expected βs
The second approach is very similar to the previous one, but in this case, steps 2 and 4 are merged into one, solving a multi-stage stochastic model including **all** scenarios from the beginning (alongside with non-anticipativity constraints). Thus, by the end of the second step we will have the optimal **expected** harvest plan for the entire scenario tree, in contrast to the previous approach where each scenario is solved separately.

The main difficult with this approach consists in the size of the extended formulation of the problem (due to number of scenarios and non-anticipativity constraints). Thus, a scenario decomposition algorithm like Progressive Hedging can be very useful for testing purposes.

C.P.: I'm testing a second approach for the main heuristic, where instead of using ignition and spreads theoretical probabilities, I'm using empirical probabilities (from simulations) inside the heuristic objective function. Current status: Testing

6.0.4 Python Implementation

Heuristic is added as an independent class within the simulator, with the possibility of use it or not when performing a simulation. When using the heuristic, simulator will record results for both approaches (with and without heuristic harvest plan), obtaining statistics for them as outputs where a comparison within number of available, harvested and burnt cells is performed.

A two sample t-test is performed (95%) in order to determine is there is a significant difference between results when heuristic is applied.

Heuristic objects allows the simulator to perform different calculations in order to compute the "value" of each cell, taking into account its economic benefits plus their potential fire risk.

1. **Heuristic Class**
   Description.
   Obj = $Heuristic(\overrightarrow{\beta})$
   (a) **Parameters**
   Inmutables
   - $\overrightarrow{\beta}$: weight's vector for value function. (array[float])

- $Price_t$: current market price $\frac{[USD]}{m^3}$. (float)
- δ: yearly discount rate [%]. (float)
- ϵ: threshold value for harvesting. (float)
- $Adjacents_i$: Number of adjacent cells for Cell i. (int)

Mutables

- $\mathbb{P}(ignition)_{i,t}$: Ignition probability of cell i for period t. (float)
- $\mathbb{P}(Spread\ to)_{i,t}$: "Spread to" probability of cell i during period t. (float)
- $\mathbb{P}(Spread\ from)_{i,t}$: "Spread from" probability of cell i during period t. (float)
- $AvailAdjacents_{i,t}$: Number of available adjacent cells for Cell i in period t. (int)
- $ProAvailAdjacents_{i,t}$: Number of available adjacent cells in expected wind's direction for Cell i in period t. (int)
- $AgainstAvailAdjacents_{i,t}$: Number of available adjacent cells against expected wind's direction for Cell i in period t. (int)
- $Benefit_{i,t}$: Economic benefits when Cell i is harvested in period t. (float)
- $Costs_{i,t}$: Economic costs when Cell i is harvested in period t. (float)

(b) **Functions & Methods**

- $AdjacentSets(Cells\_Obj, AvailCells\_Set)$:
  Updates the adjacent cells available set for cell i. (returns set)
- $ProbIgnition(Cells\_Obj, Weather, period)$:
  Computes the ignition probability for cell i in period t. (returns float)
- $ProbSpreadTo(Cells\_Obj, Cells\_Objs, Weather, period)$:
  Computes the "Spread to" probability for cell i in period t. (returns float)
- $ProbSpreadFrom(Cells\_Obj, Cells\_Objs, AvailCells\_Set, Weather, period)$:
  Computes the "Spread from" probability for cell i in period t. (returns float)
- $Adjacents(Cells\_Obj)$:
  Returns the number of adjacent cells for cell i. (returns int)
- $AvailAdj(Cells\_Obj, AvailCells\_Set)$:
  Returns the number of adjacent cells available for cell i in period t. (returns int)
- $AvailAdjAgainst(Cells\_Obj, AvailCells\_Set, Weather)$:
  Returns the number of adjacent cells available against wind's direction for cell i in period t. (returns int)
- $AvailAdjPro(Cells\_Obj, AvailCells\_Set, Weather)$:
  Returns the number of adjacent cells available in wind's direction for cell i in period t. (returns int)
- $CellVal(Cells\_Obj, Cells\_Objs, AvailCells\_Set, Weather, period, NCells)$:
  Computes the value for cell i in period t to be used in the heuristic ranking. (returns float)

6.0.5 Results

In this section we present the main results from the simulations. For each instance, we run the simulator and we record the forest's state after each year (fire periods in between), comparing its natural condition and when the heuristic is applied.

- **Test Instances**
  As a first experiment, we create small instances with 9, 16, 25, 100 and 10,000 cells. These instances are used as inputs for the simulator and harvest heuristics are applied in all of them. Each instance has a square shape (N $\times$ N) and its characteristics are summarized in **Tables 1 and 2**. When simulation starts, instances have all their cells available (no previous fires or harvest).

  1. **3 x 3 cells**
     As a first test with 9 cells instance, we can see in **Table 7** how when using any heuristic approach, about 86% of the total forest is protected (available or harvested cells) and only 14% of it is burnt by the end of the planning horizon. On the other side, about a 57.6% of the forest is lost due to the fire when no harvesting decisions (and therefore, the possibility of locating firewalls) are taken.

Statistics		**Heuristic 1**	**Heuristic 2**	**No Heuristic**
Final Status	Not Burnt Cells (AVG)	7.7617		3.8108
	Burnt Cells (AVG)	1.2383		5.1892
	Burnt cells (min,max)	(0,2)		(1,8)
Fire relative frequency [%] (Empirical probability of having n burnt cells by the end of the simulation)	0 Cells	0.0333		0.0
	1 Cell	0.2716		0.0057
	2 Cells	0.6951		0.0293
	3 Cells	-		0.0426
	4 Cells	-		0.1086
	5 Cells	-		0.4598
	6 Cells	-		0.2351
	7 Cells	-		0.2351
	8 Cells	-		0.0208
	9 Cells	-	-	
T-student Test	P-Value (95%)	0.0*	0.0*	-
Harvest results	Economic Benefit [USD]			-
	Amount Harvested [m^3]			-

Table 7: Heuristics vs No heuristic results. * indicates that the difference is statistically significant.

Based on the results, two burnt cells is the most probable result (about 70% of the cases) by the end of the simulation when applying the first heuristic, similar situation happens with the second heuristic approach. When no harvesting decisions are taken, the most probable outputs are 5 (46% of the cases), 6 and 7 (23.5% of the cases) burnt cells.

This great difference reflects the fact that the heuristic is trying to obtain the maximum economic benefit from the forest and in order to do that, it must "anticipate" and consider which cells have greater fire potential and try to harvest them earlier, protecting the rest of the forest from future fires. Thus, it will harvest more cells in the future with less risk of losing (due to fire's dynamics) them in next fire seasons. When looking at the fire relative frequency per cells in **Table 3, section 4.3** and the average heuristic solutions **Table 8**, it is possible to check this pattern. Heuristics tend to harvest earlier those cells with higher fire risk due to weather and spatial conditions, decreasing the overall fire risk within the forest.

Important is to note that heuristics' results are very sensitive to the lower demand bound per period. If we have instances with higher demands bounds, the heuristic will harvest more per period, thus protecting more area from future fires. The opposite pattern will appear when demand tends to zero.
C.P.: We need to test and apply the heuristic for a wide range of possible demands in order to adjust its weights and harvest threshold for a more general case.

As an example, we can see that cell **X** is the one with the highest empirical probability of being burnt by the end of the simulation and both heuristics tend to harvest its adjacent cells during the first periods. Important is to note that maybe they are not harvesting it, but their adjacent cells, because their appraisal could be better due to economic benefits or other factors (**see section X**). Similar patterns can be seen in the rest of the cells when looking at the heuristics solutions.

Statistics		**Heuristic 1**	**Heuristic 2**
AVG Harvesting period for Cells	Cell 1	3.34	
	Cell 2	1.00	
	Cell 3	3.87	
	Cell 4	0.00	
	Cell 5	0.00	
	Cell 6	2.92	
	Cell 7	2.00	
	Cell 8	0.00	
	Cell 9	4.00	

Table 8: Average harvesting periods for each cell using heuristic 1 and 2.

As a first approach, it is possible to conclude that **heuristic X (C.P.: to**

be done) performance is the best one in terms of total cells protected and economic benefits by the end of the planning horizon.

2. **4 x 4 cells**
   Similar analysis will be done here.

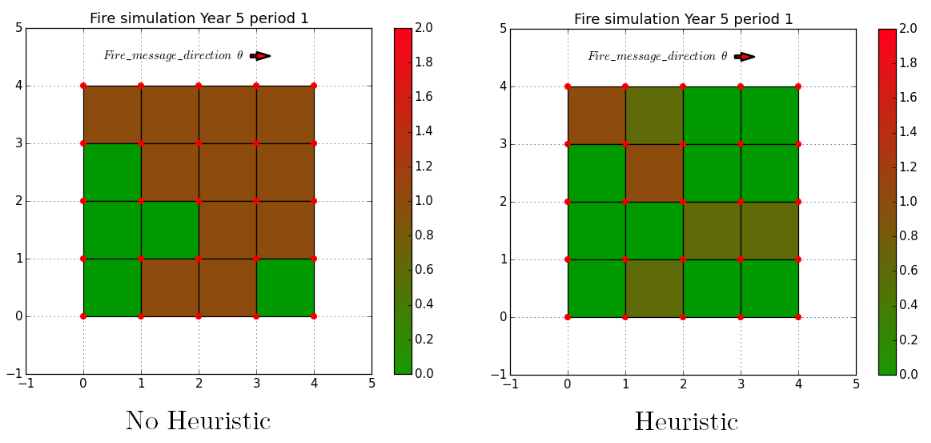


Figure 12: No Heuristic vs Heuristic results for 4 x 4 instance.

   As an example, we can see in **Figure 12** how the heuristic tends to anticipate the possible fires, based on cells' potential fire risk. In this case, instead of eleven burnt cells, only two cells got burnt by the end of the planning horizon thanks to the heuristic application.

3. **5 x 5 cells**
   Similar analysis will be done here.

4. **10 x 10 cells**
   Similar analysis will be done here.

5. **14 x 14 cells**
   Similar analysis will be done here.

- **Instance 1: Chile**
  Similar analysis will be done here.

- **Instance 2: Portugal**
  Similar analysis will be done here.

- **Instance 3: Canada?**
  Similar analysis will be done here.

	Instance	3 x 3	4 x 4	5 x 5	10 x 10	100 x 100
Heuristic 1	Available Cells					
	Burnt Cells					
	Harvested Cells					
Heuristic 2	Available Cells					
	Burnt Cells					
	Harvested Cells					
No Heuristic	Available Cells					
	Burnt Cells					

Table 9: Heuristics results summary (test instances).

	Instance	Chile	Portugal 1	Portugal 2	Canada 1	Canada 2
Heuristic 1	Available Cells					
	Burnt Cells					
	Harvested Cells					
Heuristic 2	Available Cells					
	Burnt Cells					
	Harvested Cells					
No Heuristic	Available Cells					
	Burnt Cells					

Table 10: Heuristics results summary (real instances).

7 Appendix B

Stochastic Multi-stage model: for testing purposes and future implementation.

In order to be consistent with the formulation we should compute FS and FI parameters from the forest while we are simulating. This is the model I showed to A.W. and D.M. in Chile, with the last enhancements.

I will test its performance with every new instance obtained from the simulator and with small instances I created.

C.P.: I tested it, I will add results in the next version of this document.

7.0.1 Model

1. **Sets**

   - $HPeriods\{t_h\}$
   - $FPeriods\{t_f\}$
   - $Cells\{i\}$
   - $A_i := \{j \in Cells : j \text{ is adjacent to } i\}$

2. **Variables**

   - $H_{i,t_h,s} = 1$ if cell i is harvested during $HPeriod\ t_h$ in scenario s
   - $Y_{i,t_f,s} = 1$ if cell i is burnt during $FPeriod\ t_f$ in scenario s

3. **Parameters**

   - $FI_{i,t_f,s} = 1$ if fire ignites in cell i during $FPeriod\ t_f$ in scenario s
   - $FS_{j,i,t_f,s} = 1$ if fire spreads from cell j to i during $FPeriod\ t_f$ in scenario s
   - $R_{j,i,t,s} =$ Rate of spread from cell j to i in period t in scenario s

   Where $FS_{j,i,t_f,s} = 1$ if $\frac{D_{i,j}}{R_{j,i,t,s}} < t_f$

4. **Formulation**
   max or min $z = \mathbb{E}(\text{Benefit or Cost})$
   s.t.

   (a) $\sum_{t_h} H_{i,t_h,s} \leq 1 \quad \forall i \in Cells, \forall s \in Scenarios$

   (b) $H_{i,t_h,s} \leq 1 - \sum_{\theta < t_h} \frac{Y_{i,\theta,s}}{M} \quad \forall i \in Cells, \forall t_h \in HPeriods, \forall s \in Scenarios$

   (c) $1 - \sum_{\delta \leq t_h} H_{i,\delta,s} \geq \sum_{\theta > t_h} Y_{i,\theta,s} \quad \forall i \in Cells, \forall t_h \in HPeriods, \forall s \in Scenarios$*

   (d) $\sum_{t_f} Y_{i,t_f,s} \leq 1 \quad \forall i \in Cells, \forall s \in Scenarios$*

   (e) $Y_{i,t_f,s} \geq FI_{i,t_f,s} - \sum_{\theta < t_f} H_{i,\theta,s} \quad \forall i \in Cells, \forall t_f \in FPeriods : t_f = t_h + 1, \forall s \in Scenarios$

   (f) $Y_{i,t_f,s} \geq \sum_{j \in A_i} FS_{j,i,t_f,s} Y_{j,t_f-1,s} + FI_{i,t_f,s} - M \sum_{\delta < t_f} H_{i,\delta,s}$

   $\forall i \in Cells, \forall t_f \in FPeriods : (t_f - 1) \in FPeriods, \forall s \in Scenarios$

   (g) Non-Anticipativity

   (h) $\sum_i H_{i,t_h,s} \leq (|Cells| - \sum_{\delta < t_h, i} H_{i,\delta,s} - \sum_{\theta < t_h, i} Y_{i,\theta,s})\alpha$

   $\forall t_h \in HPeriods, \forall s \in Scenarios$**

   (i) $H_{i,t_h,s} \in \{0,1\} \quad \forall i \in Cells, \forall t_h \in HPeriods, \forall s \in Scenarios$

   (j) $Y_{i,t_f,s} \in \{0,1\} \quad \forall i \in Cells, \forall t_f \in HPeriods, \forall s \in Scenarios$

(*) Constraints that are not necessary, but useful to achieve better performance of the model.

(**) Extra constraints that can be added to the model in different ways, the one shown is just an example.

5. **Explanation**
   Constraints (a) to (c) are represent harvest decisions. In (a), each cell/stand can be harvested at most once during the entire planning horizon, if we harvest a cell. The idea is that a stand will be protected from fire for the rest of the planning horizon if we harvest it and if we don't harvest a cell, it will be unprotected for the next fire period. In equations (b) and (c), if a cell is burnt, then it cannot be harvested in the future and (extra cut *) it cannot be burnt if we previously harvested it. Important to note is that this last constraint is not essential - see fire dynamic constraints - but it is a valid (and good) cut for the model, obtaining better performance.

   The second set of constraints (d)-(f) includes that every cell can be burnt at most once during the planning horizon, the fire ignition points per periods and the fire's spread dynamic, where a cell can be burnt - if it has not been harvested yet - because of a fire ignition or fire that arrives from an adjacent cell.

   Then we have the set of Non-Anticipativity constraints.

   In "Extra Constraints" (h) we can add any relationship that we want to include in the model, as an example I wrote a "Max percentage of the remaining forest that can be harvested per HPeriod", where alpha (parameter) represents that percentage. We may want to include a "lower demand bound" constraint per period or something similar.

   Objective Function can be something like:
   max $\mathbb{E}(\text{Benefits}) = \sum_s p_s (\sum_{i,t_h} U_{i,t_h,s} H_{i,t_h,s} - \sum_{i,t_f} V_{i,t_f,s} Y_{i,t_f,s})$

6. **Scenarios C.P.: Description about the scenarios and scenario trees with its parameters. Examples.**

   In this section, we describe the content of each fire scenario obtained from the simulator and used as an input for the stochastic model and heuristic.

   - **Description**
     Based on the simulator's output, a scenario is composed by two main parameters: FI and FS. These parameters summarize all the information of a fire season, including ignition points (cells that got burnt due to a lightning) and fire's dynamic (fire spread evolution from the ignition points to the rest of the forest). Then, a simulated fire can be represented from its origin to its end by these parameters.

     As an example (see **Figure 13**), suppose that we have a 3x3 forest. An ignition occurs in the cell 1 during the first fire period $t_f = 1$ ($t_f \in FPeriods$) and then the fire spreads from the cell 1 to 2 during the second $t_f = 2$ and in the next fire period from the cell 2 to 6, due to weather conditions (wind direction, speed, etc.) Then, fire stops and no more ignitions occur during the planning horizon. This scenario can be represented by a combination of FI and FS parameters as follows (where s represents the scenario s under study):

(a) **Ignitions** (FIs)
Scenario s only has one ignition in cell 1 during $t_f = 1$, then we have:
$FI_{1,1,s} = 1$ $FI_{i,t_f,s} = 0$ $\forall i \in Cells, \forall t_f : i \neq 1$ and $t_f \neq 1$

(b) **Fire Spread** (FSs)
Fire spreads from cell 1 to cell 2 in the second fire period and then from cell 2 to cell 6 in the third one:

$FS_{1,2,2,s} = 1$
$FS_{2,6,3,s} = 1$

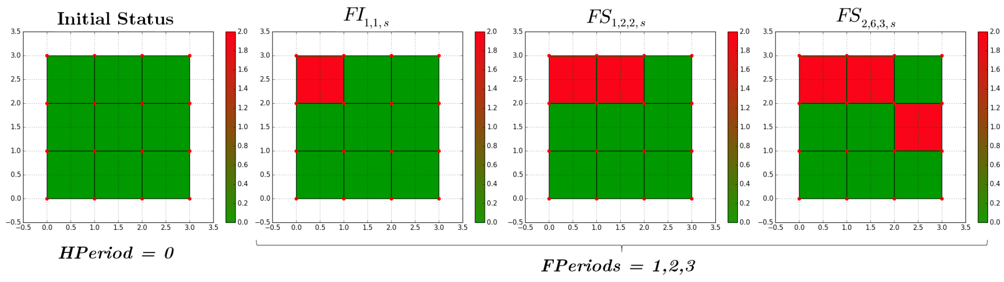


Figure 13: Example of a 4 periods scenario using the information from FI and FS parameters.

Then, using the information from FI and FS it is easy and natural to reconstruct the fire's dynamic of the scenario s.

Important is to note that the same scheme can be applied iteratively in order to obtain the entire scenario tree. As an example, we can see a scenario tree with 3 scenarios in **Figure 14**. In the initial period ($t_h = 0$) all the scenarios share the same information and forest' status (in this case, all the forest is available). Then, cell 2 ignites in scenario 1 ($FI_{2,1,1} = 1$) and cell 3 ignites in scenarios 2 and 3 ($FI_{3,2,2} = FI_{3,2,3} = 1$). Finally, fire spreads from cell 2 to 5 in scenario 1 ($FS_{2,5,3,1} = 1$), from cell 3 to 6 in scenario 2 ($FS_{3,6,3,2} = 1$) and nothing occurs in scenario 3 ($FS_{i,j,3,3} = 0$ $\forall i, j \in Cells : i \neq j$).

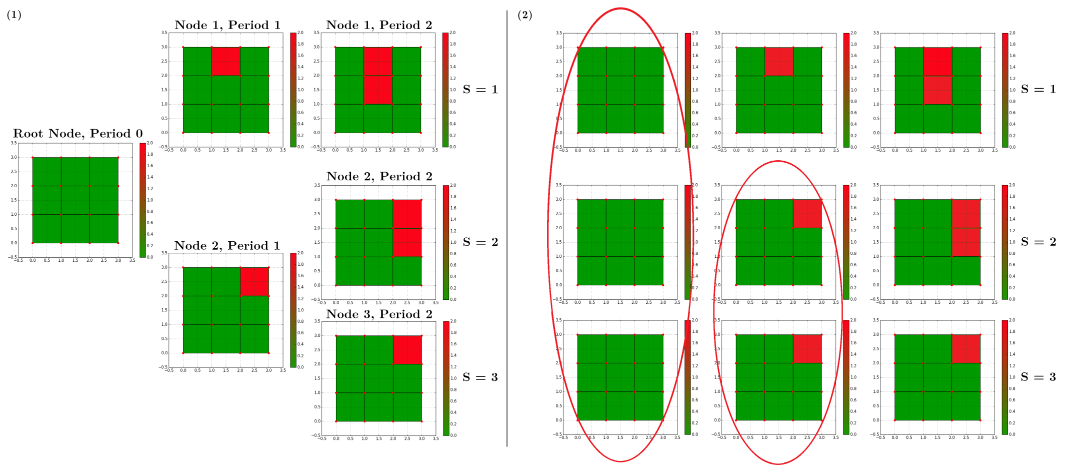


Figure 14: Example of a 3 periods scenario tree with 3 scenarios. In (1) the condensed scenario tree is showed. On the right side (2), the extended version of the scenario tree is represented, where information sets are surrounded by ellipses, representing the non-anticipativity constraints.

Using this approach, we can represent any scenario tree, with any number of scenarios and periods (multi-stage problem). Alongside with FI and FS parameters, $HPeriods$ and $FPeriods$ sets must be consistent in order to represent the entire scenario tree because some scenarios may have fewer or more $FPeriods$ than others, situation that cannot happen with $HPeriods$ since they must be the same for all scenarios because harvest decisions will be taken at the beginning of the fire season in all of them.

For example (see **Figure 15**), we have two scenarios (1 and 2) that share all their information until the second period where scenario 1 has another $FPeriod$ while scenario 2 not. In this case, scenario 1 will have an extra period in this set. In order to be consistent with the scenario tree structure, we need to take the longest $FPeriod$ set for the scenario tree and add as many artificial $FPeriods$ to all the rest of scenarios (the second in this example), copying their last status. Finally, we will have a proper scenario tree with the same number of $FPeriods$ for all the scenarios, avoiding inconsistencies in the data.

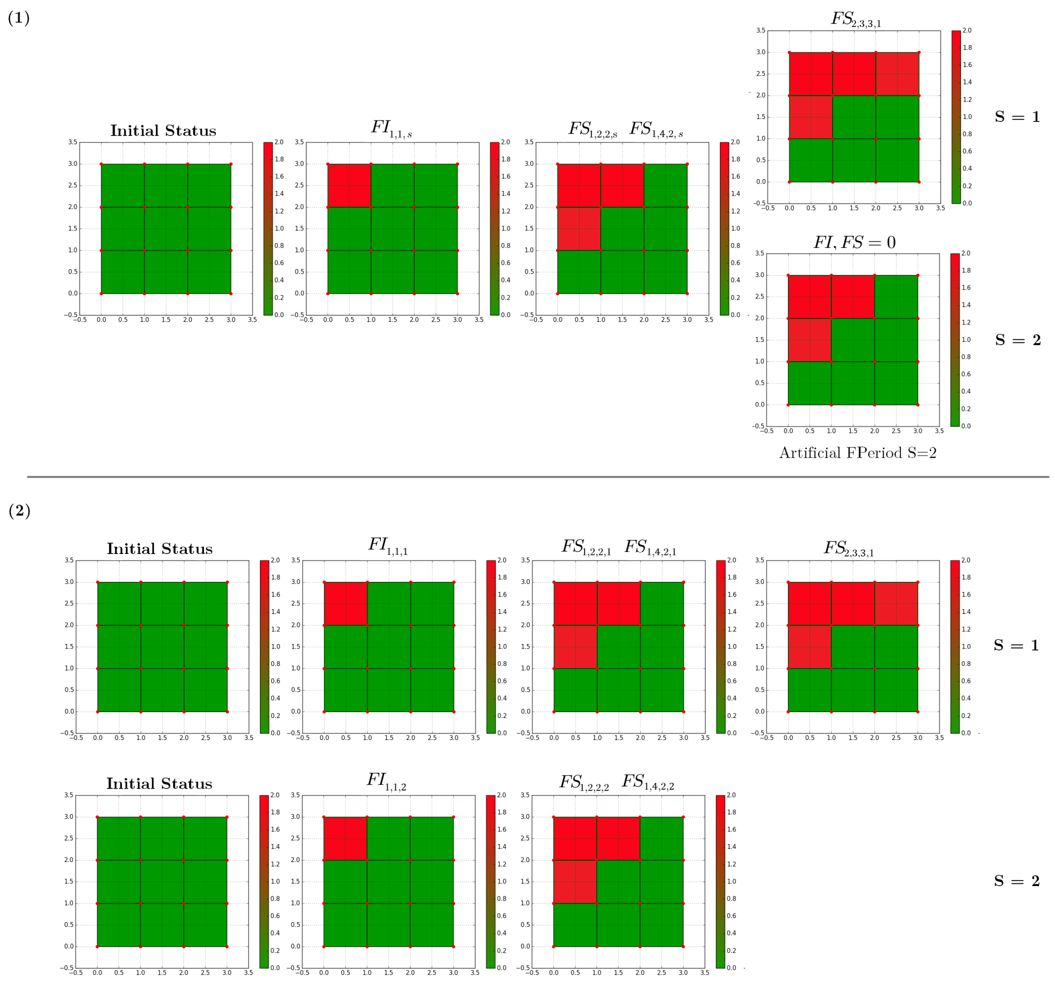


Figure 15: When two scenarios have different $FPeriods$ sets, the largest set (with higher cardinality) is selected and all the rest of scenarios are adapted to its length in order to avoid time inconsistencies within the scenario tree. The final scenario tree (1) is an adaptation of the original scenarios. An artificial $FPeriod$ is added in scenario 2 in order to have the same number of periods of scenario 1.

Thus, the complete description of any scenario and scenario trees is based on $HPeriods$, $FPeriods$ (sets), FI and FS (parameters).

- **Generation**
  All scenarios are generated from the simulator. At the beginning of each simulation, the user inputs the number of runs he wants from the simulator, generating a new scenario with each one.

  After simulating each fire season, all the fire information is translated into FI and FS parameters, generating a data file ready to be used by any commercial solver. Scenario probabilities and a scenario tree structure file are also computed and generated after all runs, determining which scenarios share information per periods or not, information that is needed for the non-anticipativity constraints.

- **Reduction**
  The original aim is to represent a wide range of fires in order to cover "all possibilities in the stochastic formulation (EF or compact model). In practice, this can lead to very large and complex models that can not be solved by traditional optimization techniques or commercial solvers. Due to the non-anticipativity constraints, the complexity of the model will grow exponential when more scenarios are included. As an example, for a forest with 4 Cells (2x2) and multiple harvest and fire periods we have thousands of possible combinations (different ignition points, spread dynamic, etc.). Taking into account that a typical forest has hundreds and thousands of cells, this situation is intractable in practice.

  Since we are dealing with a high combinatorial problem, it is needed to determine a way to reduce the total number of scenarios considered by the model. Thus, we seek to develop some scenario reduction techniques for our particular problem, allowing us to represent the original situation with a representative -reduced- subset of scenarios.

  We develop two main approaches to reduce the original scenario tree size to a representative subset based on the forest's cells status:

  (a) **Ignition Points**
  As a first approach, the reduction methods of this category don't take into account the fire's dynamic or the complete fire scar but only the ignition points. Since original weather conditions should be very similar for each simulation (each fire season has the same initial conditions), the fire should be spread in the same direction. Based on this point, the most relevant events for describing a scenario are the different ignition points and when these ignitions occurs ($FPeriods$) within the forest.

  Thus, two scenarios will be more similar if they share ignition points (FI) and the periods of those ignitions. On the other hand, if all their ignition points are different (both in terms of spatial distribution and periods) then they will be treated as two independent scenarios within the reduced scenario tree. Therefore, they cannot be represented by only one scenario.

  - **Method 1**
    Suppose that each cell inside the forest is represented by a square of side a. For a square with coordinates (x, y) in its bottom left vertex, its mass center will be represented by $(x+\frac{a}{2}, y+\frac{a}{2})$. Then, we can define the

euclidean distance of two cells (i, j) as $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ so we get the traditional expression.

In order to measure the difference/distance between scenarios, we need to compute the distance between ignition points for all periods of the planning horizon, obtaining the final distance metric for each pair of scenarios (i, j). Then, if the total distance is less than some threshold $\epsilon > 0$, it is possible to say that scenarios (i, j) are similar enough to be represented by one scenario. Thus, we need to compute $d_{i,j,t}$ $\forall t \in FPeriods$ with ignitions (first $FPeriod$ after a $HPeriod$ in the first version of the simulator). Finally, the total "distance between scenarios (i, j) will be:

$$d_{i,j} = \sum_{t \in FPeriods)} d_{i,j,t} = \sqrt{(x_{i,t} - x_{j,t})^2 + (y_{i,t} - y_{j,t})^2}$$

Thus, we measure the distance between ignition points for each period. Important is to note the situation when scenario i has a new fire (due to a new ignition) and scenario j not. In this case, we use the diagonal of the square since it is the longest possible distance between two points in a square. Then, for a $N \times N$ forest with cells of side a, the distance will be $Na\sqrt{2}$.
In order to select the scenario that will represent a set of them, we have different (and simple) options:

i. We can take the one with the highest probability to represent all the scenarios.

ii. Depending on the number of scenarios we want in our reduced scenario tree, we can rank them by their probabilities, characteristics or a mix of them. Then, we select the number of scenarios according to our target number.

– **Method 2**
Using the same euclidean metric from method 1, we can follow a K-means cluster algorithm scheme. In this case it is possible to input the size of the scenario tree, generating K scenario clusters represented by a centroid. In order to compute these centroids, each scenario will be represented by the coordinates of its ignition points (center of mass per period) as a vector of length $|HPeriods|$ (one scenario cannot have more than $|HPeriods|$ ignitions in the current simulator), where each component consists of a pair of coordinates $(x + a, y + a)$, representing the center of mass of the ignition point.

Once we identify the K clusters and their centroids, we compute the distance between each scenario and its cluster's centroid. Finally, using the euclidean distance, we select the scenario that is closest to its centroid as the scenario to keep in the reduced scenario tree.

Important is to take into account the presence of Black Swans (scenarios with very low probability of occurrence, but with high -usually negative- impact in the final results). Then, at least 5% of these scenarios will be kept as part of the final scenario tree, as an extra constraint for the original algorithm.

– **Method 3**
  Similar to Method 1, but in this case we will divide the forest in four sections: North-East, North-West, South-East and South-West. In this method, individual differences are not relevant and instead of taking a look at each cell, we will take into account the area of the ignition. This is helpful since we don't need to process each cell as a separate entity, with all the associated computational cost.

  Since initial weather conditions are the same for each simulation, the fire's dynamic should be very similar for ignition points that are close in the map (neighbor ignitions points should have a very similar final fire scar). Based on this, we can compare scenarios according to these four sections of the forest, computing the euclidean distance between their center of mass.

  These approach can be refined/improved including forest data and characteristics. For example, we can divide the forest in sections depending on the fuel type, the soil composition, the altitude of the terrain, and other cells' characteristics. This can be useful because the fire's dynamics are correlated to these elements, then, we can expect that a fire scar from a particular section will evolve following a certain pattern (with high probability) during the planning horizon, allowing us to reduce the number of similar scenarios.

  Again, we can select the representative scenario using the K-means centroids scheme.

All previous methods can be tested for different distance metrics. In particular, we can represent our forest with a $N \times N$ matrix, full of 0s and 1s depending on the ignition position (where 1s indicate an ignition) allowing us to use distance/similarity metrics between matrices. Moreover, an extension of this approach is to use 0s and tf values in a matrix to represent a scenario based on its ignitions, where a coordinate $a_{i,j} = t_f$ within a matrix represents an ignition during the t_f period. Then, we can compute a more accurate distance metric between scenarios.

Some distance/similarity metrics being tested:

i. **Frobenius Distance (F)**
For two square matrices (A, B), we can compute $F_{A,B}$ as follows:

$$F_{A,B} = \sqrt{trace((A - B) \times (A - B)')}$$

where B represents the conjugate transpose of B

ii.

CP: I'm testing all these metrics. I will have all results by the end of June

(b) **Fire Scar and Evolution**

- **Examples**

7. **Tests**
C.P.: Will be ready in next version of this document.
Instances.

8. **Results**
C.P.: Will be ready in next version of this document.
Results obtained.

9. **Examples**
In **Figure 14** we can see a yearly example (for testing purposes), where harvest decisions are taken every January 1st ($HPeriods$), then, a period without anything relevant occurs [1] and then the fire season ($FPeriods$) starts in June 1st [2]. During this season, ignitions and fire spread happen - in this example, only one fire occurs and it lasts 8 hours - and after it, we have another period without fires [3].

A new year starts with harvest decisions, taking into account the actual/new forest status.

This scheme can be generalized including more fires, but the main idea is to show that there is no need to use the same time scale for both time sets - $HPeriods$ and $FPeriods$ - if they are consistent.

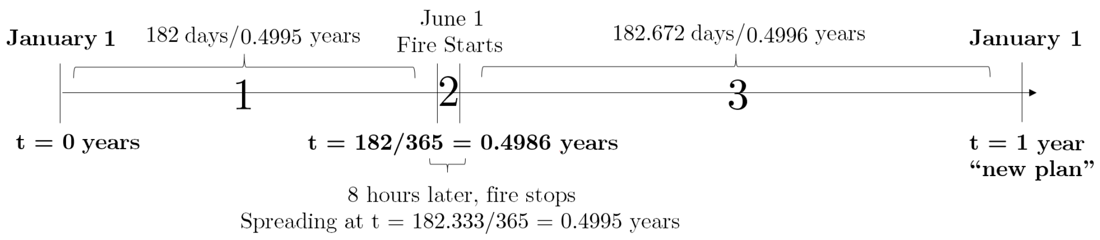

January 1
182 days/0.4995 years
June 1
Fire Starts
182.672 days/0.4996 years
January 1
1
2
3
t = 0 years
t = 182/365 = 0.4986 years
t = 1 year
"new plan"
8 hours later, fire stops
Spreading at t = 182.333/365 = 0.4995 years

Figure 14: Example of a yearly scale for SMSP (Stochastic multi-stage problem).

I will add more description later...