Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

ASYMMETRIC CELL-DEVS WILDFIRE SPREAD USING GIS DATA

Mark Murphy[1], Jaan Soulier[1], Alec Tratnik[1], and Gabriel Wainer[1]

[1]Dept. of Systems and Computer Eng., Carleton University, Ottawa, ON, CANADA

ABSTRACT

Wildfires are extremely dangerous and destructive. In order to protect populations and infrastructure, government officials and firefighters must best decide how to expend their limited resources. Wildfire simulation aids these decision makers by predicting the spread of fire. One method to simulate wildfires is using Cellular Automata. In this paper we present a hybrid model combining a cellular wildfire spread model using asymmetric Cell-DEVS models, the Behave library for calculating the rate spread of the fire (based on difference equations), and a Geographical Information System. The GIS information from publicly available maps, and the combination of techniques can improve the accuracy of the wildfire spread predictions, and allows multiple scenarios to be simulated in timely manner.

1 INTRODUCTION

Accurate forecasts of the spread of wildfires help decision makers deploy firefighting resources and plan the safe evacuation of at-risk areas. The modelling and simulation (M&S) of wildfires is extremely complex; there are many factors which affect the rate of spread including type and volume of alive and dead vegetation, moisture content, weather factors such as humidity, wind, precipitation, and terrain factors such as elevation change. Forest fire spread models, such as the US Forestry Management Behave model, have been in use for over fifty years, and are constantly improving due to updated models, particularly vegetation burn rate models known as fuel models. Wildfire spread models can only be as accurate as the data that is input, which can be challenging to gather and incorporate into a model, especially over large areas of varying terrain and vegetation. In order to quickly and accurately model active fires, or areas at risk of fire, a technique must be developed to fuse multiple sources of information into a simulation to calculate the fire's spread.

Cellular Automata (CA) have been widely used for modelling of physical systems; their components, known as cells, are usually arranged in a grid pattern and connected to each other in a defined way. Each cell is a representation of a part of the physical phenomena, and this is calculated using the influence of the states and actions of the cells surrounding it, known as its *neighborhood.* Based on this information, each cell calculates a change to its own state based on inputs from the neighborhood using a discrete time update. The Cell-Discrete Event System Specification (Cell-DEVS) formalism is a hybrid modeling technique that combines CA with discrete-event modeling, and is well suited to simulate complex problems. Cell-DEVS uses an asynchronous update method, as it is based on the DEVS formalism, as well as a discretization of the space using discrete components (Wainer 2009). Also, using the Cell-DEVS models, we can make use of a sound methodology for describing complex timing behavior without knowing the simulation mechanism of the delays. This enables the definition of complex cellular models, reducing development time related to the programming of timing control. When modelling natural phenomena, it is often not ideal to have a square grid of uniform cells, for this reason Cell-DEVS has been recently extended to allow more complex tessellations, including complex irregular grid shapes and non-uniform cells (Cardenas and Wainer 2022). Such models, called Asymmetric Cell-DEVS, are implemented in the Cadmium simulator (Belloli et al. 2019), a C++ library, which can efficiently configure scenarios using inputs from JavaScript Object Notation (JSON).

In this research, we extended the use of M&S of the spread of fire combining multiple techniques: discrete-event modeling, asymmetric tessellations, an open-source library called BehavePlus, which provides an implementation of fire spread using differential equations, terrain and vegetation inputs from publicly available maps published by Natural Resources Canada, open source mapping software, and Python libraries. A Geographical Information System is added to extract information from the maps, and to select a simulation region and build individual cells for each area of landmass, including elevation and vegetation types. The information is input into a Cell-DEVS model simulated using the C++ Cadmium library, which is combined with BehavePlus calculations to determine the amount of time the fire will take to reach the neighboring cell. The results can be viewed in a user-friendly user interface. The combination of these techniques can improve the accuracy of the wildfire spread predictions, and allows multiple scenarios to be simulated in timely manner.

This paper is divided into 4 sections, the first being this introduction. Section 2 introduces Discrete Event Simulation Specification (DEVS) using Cellular Automata and applications for modelling fire spread as well as the fire spread model BehavePlus. Section 3 defines the forest fire model and describes the application. Section 4 is conclusion and references.

2 BACKGROUND

Wildfires are extremely complex and cannot be solved analytically due to the large number of variables and unknowns. Within the fire itself there are phenomena such as radiation, convection and diffusion; environmental factors such as wind, humidity and precipitation; and terrain factors such as slope, vegetation burn rate and vegetation volume. Due to the large number of factors, modelling the fire with theoretical mathematic models relying on heat transfer and energy conservation has proven difficult. Researchers have found success in creating mathematical models based on live fire experiments, such as (Balbi et al. 1999). Burn times and maximum temperature of the fire can also be important factors, especially in the long term under various wind speeds, methods such as (Albini and Reinhardt 1995) and (Mandel et al. 2011). A comprehensive review of the different spread models can be found in (Ning et al. 2024).

There are many models that have gained prominence around the world. In Canada, the Fire Behaviour Prediction System (Forestry Canada Fire Danger Group, 1992) was developed using observations from over 400 fires to represent fires using non-linear equations. It has modelled sixteen different fuel types, slope and wind, and a fire weather component that considers moisture, latitude and other effects called the Canadian Fire Weather Index (FWI). The latest implementation is a Government sponsored project called PROMETHEUS (Yossemi et al. 2006). In Australia, the work of (McArthur 1962) for the Forest Fire Danger Meter forms the foundation for many models. The models are based on a statistical analysis of prescribed burns in grassland and eucalyptus. While the fuel models are endemic to Australia, comparisons are still made to these models, and they are still important for Australian fire spread models (Cruz et al. 2015). The PROPOGATOR model was developed by the Italian Government, but its models have been expanded for much of Europe. It is a probabilistic spread model using high quality mapping of the terrain which utilizes a mix of many of the same factors as the other models discussed; slope, wind, fuel composition and moisture. It has been expanded to account for firefighting activities (Trucchia et al. 2020)

One of the most popular models of fire spread based on empirical results is (Rothermel 1972). The Rothermel model is a quasi-empirical model based on data from wind tunnel experiments with ignition of artificial fuel beds. The basic spread model uses conservation of energy equations for fires in the surface fuel, relying on fuel moisture and terrain data (Andrews 2018). This model is the base for all operational fire models in United States (Scott 2021). The United States Department of Agriculture's Forest Service operates multiple models such as BehavePlus, NEXUS, FARSITE and Nomographs, each offering different levels of complexity (Scott 2021). Nomographs are for field calculations when a computer is not available, quick calculations can be made to predict the range of spread rates using lookup tables for inputs of wind, slope, fuels and canopy height. BehavePlus predicts fire behavior at a point and produces detailed charts of fire shape, NEXUS specializes in the interaction between surface and canopy fires, and the more complex

FARSITE models, including the popular FlamMap application, use detailed landscape and vegetation data to calculate the path of spread for the wildfire (Andrews 2007).

In this work we used BehavePlus, which offers a mechanism for integration with the existing tools and is publicly available. BehavePlus has modules to calculate surface spread, crown spread, spotting distance, containment lines, ignition probability and fire damage estimates such as scorch area and tree mortality. BehavePlus' Surface module (Heinsch and Andrews 2003) is an amalgamation of eighteen different models, with modifications to the original Rothermel model to improve factors such as wind speed, wind affect at various heights, flame and fire line intensity, spread direction, slope corrections, multiple different types of fuel burn rates and behavior, and how to make calculations for combined fuel types. A history of these improvements is described in (Andrews 2018), including a detailed breakdown of the formulas used in the most recent version and detailed descriptions of their inputs. Improvements to the fuel models have improved the accuracy of the calculations due to increased accuracy of the individual components of the formula, research in this area continues today.

In addition to their operational use by government agencies, the BehavePlus modules are used by researchers to investigate past fires (Drury 2019), confirm accuracy of and develop new fuel models - both for vegetation types around the world (Grabner et al. 1997, McGranahan et al. 2012, Little et al. 2024) and for forest moisture, fuel height and litter depth (Glitzenstein et al. 2025, Fernandes 2009), and as the calculation engine for fire spreading simulations (Ntaimo et al. 2004, Sousa et al. 2011).

A different approach to solving the problem of wildfire spreading is the use of Cellular Automata (CA) (Wolfram 1994), a popular strategy used in modelling of forest fires (Xu 1994, Yassemi et al. 2008). CA allow a simplified way to calculate the fire spread, especially under uniform slope and terrain conditions. CA became popular for computational efficiency, taking advantage of the tradeoff between the size of the cell and the number of calculations required; if the cells are not close enough together the fire will spread with distortion (Ghisu et al. 2015). It is easiest to implement CA models in a uniform square grid as it greatly reduces the model complexity. More recent CA models are able to overcome issues with the uniform cell pattern, adding variability in cell state, and irregular sized simulation areas (Trucchia et al. 2020, Freire and DaCamara 2019).

In an ordinary CA model, each cell is updated at each time step. This is computationally inefficient, especially if there is no possible way for a cell to change state at the next time step. CA simulations can be improved using a discrete-event approach such as the Discrete Event System Specification (DEVS) formalism, with every cell represented by its own model. There have been many implementations of forest fire models with DEVS, the majority combining DEVS and cell spaces. Early DEVS implementations such as (Ntaimo et al. 2005, Dhal 2015) used the fire spread rate as an update rule, while others such as (Xue et al. 2012, Muzy et al. 2005) use a temperature calculation, such as heat flux, and burn time. One drawback of these implementations is computational complexity. Further simplifications have been implemented to speed up DEVS simulations such as splitting temperature ranges using Quantized DEVS models to reduce calculations (Al-Habashna et al. 2019) or split partitioning the simulating tasks to improve simulation time (Guo and Hu 2010). Cell-DEVS uses an asynchronous update to reduce calculation, and has been successfully implemented in (Muzy et al. 2005, Macleod et al. 2006, Al-Habashna et al. 2019). In particular, various implementations of forest fires have been built using the Cell-DEVS formalism. In order to improve computation time, the inactive cells in the Cell-DEVS models are forced into a quiescent state (Cardenas and Wainer 2022), and the cells are updated asynchronously, using the DEVS abstract simulation algorithm. This can reduce computation time, especially in a fire spread model where the primary concern is the leading edge of the fire.

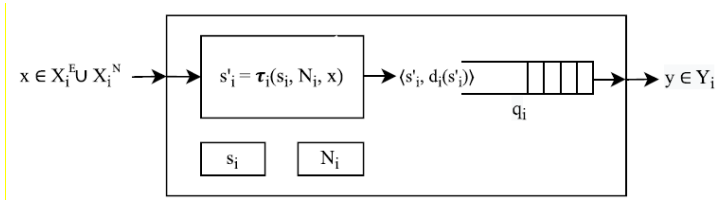

Figure 1: Schematic of a cell in the asymmetric Cell-DEVS formalism.

Cell-DEVS is an extension of DEVS that combines CA and DEVS (Wainer 2009). The Cell-DEVS atomic model defines the behavior using a local computing function to update the cell's state, and the delay function. Coupled Cell-DEVS models form the cell space lattice, including external input and output coupling with other models, the number of cells in each dimension, the neighborhood shape and a set of border cells, which can have different behavior than the interior cells or can be configured with wrap-around boundaries.

Most cellular modelling methodologies use a regular tessellation (normally, square cells). However, for the spread of natural phenomena, this is not as effective as the cell spaces can be better modelled using a more generic tessellation. For instance, in wildfire modeling, a square-grid neighborhood may be used, however the diagonal cell is at a distance of ~1.4 times the distance as the cells directly above and below and may lead to uneven spread modeling. Instead, a hexagonal (or polygonal) grid can provide better results. Therefore, we used asymmetric Cell-DEVS, where the neighboring cells are not necessarily the immediate nearby cells, and not all the neighbor cells affect the cell behavior in the same way. In our model, each cell uses different elevation level and fuel type (i.e., fire does not spread over water or barren land).

In order to define our models, different software tools were needed. First, we used a Geographical Information System (GIS), a piece of software composed of data and software used to deal with spatial information on a computer. With a GIS application the user can view digital maps, add spatial information and perform spatial analysis. One popular free and open-source application is called QGIS (QGIS Developmental Team, 2021). QGIS allows users to create custom plug-ins to manipulate map data using custom scripts programmed in Python. The QGIS software can read many types of maps; this example implementation uses freely available elevation and landcover raster maps published by the Canadian Department of Natural Resources in the .tif format.

To build the simulation models we used Cadmium, a header-only library written in C++ that allows the modeling and simulation of computational models based on the DEVS and Cell-DEVS formalisms. In Cadmium, cells are implemented in C++ and the cell space is defined using a JSON configuration file. The JSON file has an entry for each cell, specifying its neighbors by name, and thus the cells can be connected in any way the user would like. This approach allows us to study multiple setups by simply modifying the configuration file, thus avoiding recompilations and reducing the overall time required for exploring a scenario. Furthermore, modelers can integrate Cell-DEVS models with other DEVS models implemented with Cadmium.

Finally, we make use of the library BehavePlus to calculate the spread rate. In this case, fire modules are available as a standalone application, but have also been implemented as a C++ library (RMRS Missoula Fire Sciences Lab), the asymmetric Cell-DEVS model described in this paper calls on the Behave library to calculate fire spread times between cells. The Surface module takes the inputs of wind speed and direction, slope steepness, five different moisture factors, canopy height, canopy coverage percentage and the fuel model; areas with multiple types of fuel use a blended result. The Surface Module will calculate the spread of the fire in all directions along with a flame length.

In the following sections we discuss how these methods and tools were used to integrate forest fire

3 DEFINING A FOREST FIRE MODEL

The forest fire model being presented allows for rapid development of simulation scenarios for different simulation areas, initial fire sizes, start times and fidelity of the model. Figure 2 (Cardenas and Wainer

2022) depicts the process to build a model and simulate a wildfire. To build the model geographical data is required, a Cell-DEVS generator will use the geographical information and user selected scenario parameters for scenario configuration, outputting a JSON simulation file. This file contains an asymmetric Cell-DEVS scenario that is run on the Cadmium simulator, producing a simulation results in the form of log file which can then be viewed using GIS visualization software.

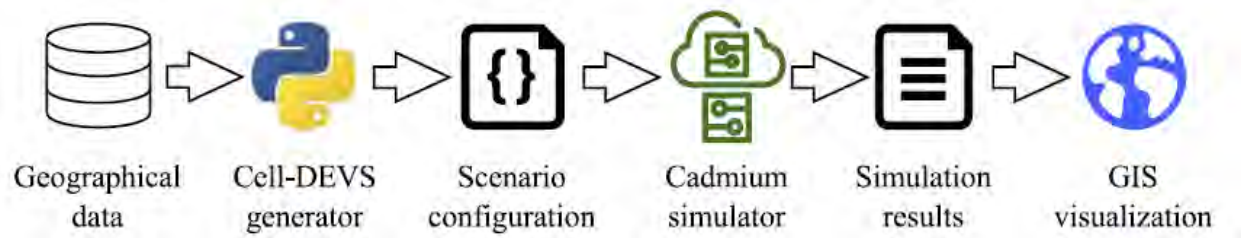

Figure 2: Cell-DEVS M&S process using GIS data.

The Cell-DEVS generator uses QGIS to extract GIS data from a map representing the area of interest and merge it into the scenario. The generator is a custom plug-in that combines QGIS's drawing and visualizations tools with the flexibility of Python scripts. The plug-in takes the user selected simulation regions and cell size and returns a JSON file with scenario configuration. The Python library Rasterio creates a mask of the simulation areas and resizes the maps to those areas. It then creates a hexagonal grid and samples the map for the relevant GIS data, including which of the neighbor cells are in simulation region. The generator takes the user selected simulation regions, wind parameters and desired cell size and returns a JSON file with scenario configuration. The Python library Rasterio creates a mask of the two simulation regions and uses it to reduce the size of the map's layers to the relevant area. Each GIS map has a key called a transform that translates a position on a map to a position within a known coordinate reference system. A new layer for the ignition region is created using the mask layer, it contains information regarding whether that area is on fire or not at the beginning of the scenario; the result is a binary layer in the same coordinate format as the elevation layer. The landcover region is then resampled to match the transform of the elevation layer. The generator then starts in the bottom left of the elevation layer and creates a hexagonal grid, it checks each point to see if there is relevant data, meaning it is in the simulation region. If it is in the simulation region it samples for the relevant GIS data, including which of the neighbor cells are in simulation region. It applies the transform to determine the coordinates of the cell and its neighbors, and writes all of the data into the JSON scenario configuration file.

This capability is demonstrated in Figure 3 for two regions, the simulation region in yellow, and the ignition region in red. Figure 3 shows an elevation layer map (on the right) and a landcover layer map (on the left) uploaded to QGIS. The landcover layer shows different vegetation types and features such as rivers, lakes and rock. The elevation layer represents different elevations (seen as different shades). The landcover layer is used to determine the fuel type, and the elevation layer (obtained from Natural Resources Canada) is used for the cell's elevation in order to calculate the slope between cells. We built a method to align the layers but only one can be viewed at a time. The plug-in gives the ability to configure the scenario by selecting the area, ignition region, resolution, the wind speed and direction. This data is generated for each cell, along with its neighborhood, and written to a scenario file, defined using JSON.

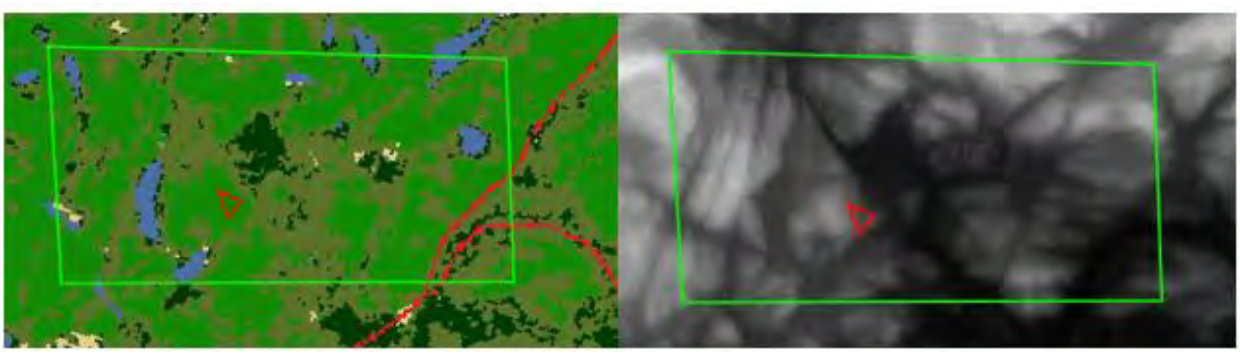

Figure 3: Landcover layer (left) and elevation layer (right) with yellow and red polygons representing the simulation area and the ignition area.

The resulting JSON representation for each cell is shown in Figure 4. Cell names are represented as the location on the map in meters referenced to a fixed point defined in the map's parameters. The other cells in the neighborhood are listed, along with their distance away, so that simulator knows which cells influence its behavior, it can be seen that the cells to the left and right of this cell are a distance of 10m away, and the ones offset in the rows above and below are 11.18m away. This cell has a fuel of Type 8 based on the data retrieved from the landcover map and indicated by *fuelModelNumber*, an *elevation* of 325.92m was retrieved from the elevation map, and it is not "ignited" as this point was not in the polygon selected for the region initially on fire at the start of the simulation.

```
"470990_5087210": {
            "neighborhood": {
                "470990_5087210": 0, "470995_5087200": 11.18,
                "470985_5087200": 11.18,  "470980_5087210": 10  },
            "state": {
                "fuelModelNumber": 8,"windDirection": 180,
                "windSpeed": 11,         "x": 470990.0,
                "y": 5087210.0,          "elevation": 325.92,
                "ignited": false                                    }
        },
```

Figure 4: Example of a cell output to a JSON file for use in the Cadmium simulator.

The Cadmium simulator uses this JSON scenario above to create a coupled model linking each of the cells to all of its neighbors, creating the necessary input and output ports between the two as well as initial conditions. The state of each cell includes its *x* and *y* coordinates on the map, distance calculations and for logging, *ignited* - whether the cell has caught fire or not, *willIgnite* – becomes true when a neighboring cell is on fire and it is calculated that it will spread to the current cell, *ignitionTime* – the time that the fire will reach the midpoint of the cell based on calculated spread rate, *spreadDir* – the direction that the fire is coming from, *rateOfSpread* – calculated rate of spread from the neighboring cell.

The next step is the definition of the atomic Cell-DEVS models using the Cadmium tool. Figure 5 shows a code snippet in which we can see the behavior of each cell in the cell space.

```
class Cell : public cadmium::celldevs::AsymmCell<State, double> {

 public:
  Cell(const std::string& id, const std::shared_ptr<const
        cadmium::celldevs::AsymmCellConfig<State, double>>& config)
            : cadmium::celldevs::AsymmCell<State, double>(id, config) {}

  State localComputation(State state, const std::unordered_map<std::string,cadmium::
            celldevs::NeighborData<State, double>>& neighborhood) const override  {

    //If cell already ignited, it has already told its neighbors and it can passivate
    if (state.ignited)   {
```

```
      state.sigma = std::numeric_limits<double>::infinity();
      return state;
      }

    //When it reaches ignition time needs to change to ignited and tell its neighbors immediately
    if (state.ignitionTime <= this->clock) {
       state.ignited = true;
       state.sigma = 0.0;
       return state;
      }

    //if not ignited check if it will ignite, or ignite faster than previously calculated
    for (const auto& [neighborId, neighborData]: neighborhood)  {
        //check if the neighbor is ignited, if not go to the next neighbor
        if (!neighborData.state->ignited) continue;

        surface.doSurfaceRunInDirectionOfInterest(direction,
                  SurfaceFireSpreadDirectionMode::FromIgnitionPoint);

        spreadRate = surface.getSpreadRateInDirectionOfInterest (SpeedUnits::MetersPerSecond);

        //If the spread rate is not high enough, ignore it
       if (spreadRate < DBL_EPSILON || spreadRate != spreadRate) continue;

       //Calculate how long it will take to spread to this cell
       const double timeToWait = distance_btwn / spreadRate;

       //If fire spreads from another cell faster, keep that ignition time
       if(state.ignitionTime < this->clock + timeToWait)  {
                state.sigma = std::max(state.ignitionTime - this->clock, 0.0);
                continue;
       }

        //calculate new ignition time, set the spread rate and direction
       state.ignitionTime =  this->clock + timeToWait;
       state.rateOfSpread = spreadRate;

       //make the direction more readable for the log
       if (direction<0)  direction = 360+direction;

       state.spreadDir = (int)direction;

       //set the cell to waiting to ignite mode
       state.willIgnite = true;

       //passivate until ignition time
       state.sigma = std::max(state.ignitionTime - this->clock, 0.0);
       }

      return state;
    }

    double outputDelay(const State& state) const override { // delay function
        return state.sigma;
    }
};
```

Figure 5: Cell update rules.

Under the rules of asymmetric Cell-DEVS, the cell will only update its state at the beginning of the simulation, when awakened by one of its neighbors, or when it is time to ignite. All cells will start with the delay *sigma* set at infinity, the quiescent state. When it comes time for the cell to update its state, it will first check if it has already ignited, and if so will stop and return to the quiescent state. If it has not ignited it will check if it is time to ignite; if it is time to ignite it will set the delay *sigma* to zero in order to immediately inform its neighbors and update its state that it is *ignited*.

For a cell that has not reached its ignition time it will check all of its neighbors to see if they are on fire, a cell that has no neighbors on fire returns to the quiescent state. If there is a neighbor on fire it will calculate the spread rate for the fire by calling the BehavePlus library, by using the *surface* methods. Using the spread rate and distance between cells, the ignition time for the cell can be calculated. If there are more than one neighbor on fire the soonest ignition time is used. The cell sets the delay *sigma* to wake the cell up at ignition time. The updated state and the output delay are returned to the coupled Cell-DEVS model.

Before calculating the spread rate from the neighbor cell, the parameters for the BehavePlus model must be loaded, as shown in Figure 6. Using the location and elevation of the cell and its neighbor, the direction, distance and elevation change and aspect can be calculated. The wind speed and direction are taken from the JSON. The model runs the calculations twice, one for each fuel model, and averages them. the two fuel models are for the current cell and the neighbor cell. There are additional parameters that use realistic values, in the future these parameters can be incorporated into the Cell-DEVS generator; these include five moisture readings, canopy and crown cover, and canopy height.

```
//calculate neighbour distance, direction and elevation change relative to current cell
deltaX = state.x - neighborData.state->x;
deltaY = state.y - neighborData.state->y;
direction = atan2(deltaX, deltaY) * 180.0 / M_PI;
distance_btwn = sqrt(pow(deltaX,2) + pow(deltaY,2));
elevation_change = state.elevation - neighborData.state->elevation;
// set aspect for which direction between the two center points is uphill
if (elevation_change < 0 {
    slope = -100*elevation_change/distance_btwn;
    aspect = direction + 180;
}
else {
    slope = 100*elevation_change/distance_btwn;
    aspect = direction;
}
//Set parameters for the BehavePlus model
FuelModels fuelModels;
Surface surface(fuelModels);
surface.updateSurfaceInputsForTwoFuelModels(
    state.fuelModelNumber,// firstFuelModelNumber
    neighborData.state->fuelModelNumber, // secondFuelModelNumber
    5.0, // moistureOneHour
    6.0, // moistureTenHour
    7.0, // moistureHundredHour
    30.0, // moistureLiveHerbaceous
    30.0, // moistureLiveWoody,
    FractionUnits::Percent, // moistureUnits
    state.windSpeed,   //windSpeed
    SpeedUnits::KilometersPerHour, // windSpeedUnits
    WindHeightInputMode::DirectMidflame, // windHeightInputMode
    state.windDirection,  // windDirection
    WindAndSpreadOrientationMode::RelativeToNorth, // windAndSpreadOrientationMode
    50.0, // firstFuelModelCoverage,
    FractionUnits::Percent, // firstFuelModelCoverageUnits
    TwoFuelModelsMethod::Arithmetic, // twoFuelModelMethod
    slope,         // slope
    SlopeUnits::Percent, // slopeUnits
    aspect, // aspect
    30.0, // canopyCover
    FractionUnits::Percent, // canopyCoverUnits
```

```
    10.0, // canopyHeight,
    LengthUnits::Meters, // canopyHeightUnits,
    40.0, // crownRatio,
    FractionUnits::Percent // crownRatioUnits
);
```

Figure 6: Arguments for activating the Behave library.

The outputs *time, x, y* and *ignited* are used by the QGIS temporal controller to display the burn pattern using the coordinates of the cell center and the time it becomes ignited. The next four columns, *willIgnite*, *simtime*, *ignitionTime* and *sigma* are used to verify the correct functioning of the logic. *spreadDir* and *rateOfSpread* are used to determine which direction the fire spread came from and at what rate.

time	x	y	ignited	willignite	simtime	ignitionTime	sigma	spreadDir	rateOfSpread
28/04 10:30	470819	5089244	1	0	0	inf	0	0	0
28/04 10:30	470945	5089244	0	1	0	22441.4	22441.4	90	0.00561462
28/04 10:30	470882	5089118	0	1	0	23238.8	23238.8	153	0.00606194
28/04 10:30	470819	5089244	1	0	0	inf	inf	0	0
28/04 16:44	470945	5089244	1	1	22441.4	22441.4	0	90	0.00561462
28/04 16:44	470882	5089118	0	1	22441.4	23238.8	797.402	153	0.00606194
28/04 16:44	470819	5089244	1	0	22441.4	inf	inf	0	0

Figure 7: Sample logger output.

Visualization of the simulation results are made possible by Cadmium's robust logging function. For this simulation the logger is set up so that it outputs in a format that can be read by the QGIS software. QGIS has a *Temporal Controller* function that allows the user to move time forward in discrete increments and display time coded GIS data. Cadmium displays the location of the cell's midpoint and the time that it becomes ignited. This model was programmed to accept start time of the simulation as an input, allowing easy comparison of the results to real fires which occurred in the past or to make predictions for a current fire or potential future fires. The example below shows the initial conditions at the start of the fire, with the ignition polygon all ignited, it is a wide grid of approximately 2km x 4km, and a large grid spacing of 30m at twelve-hour intervals. The vegetation represented by light green squares has a much quicker spread rate.

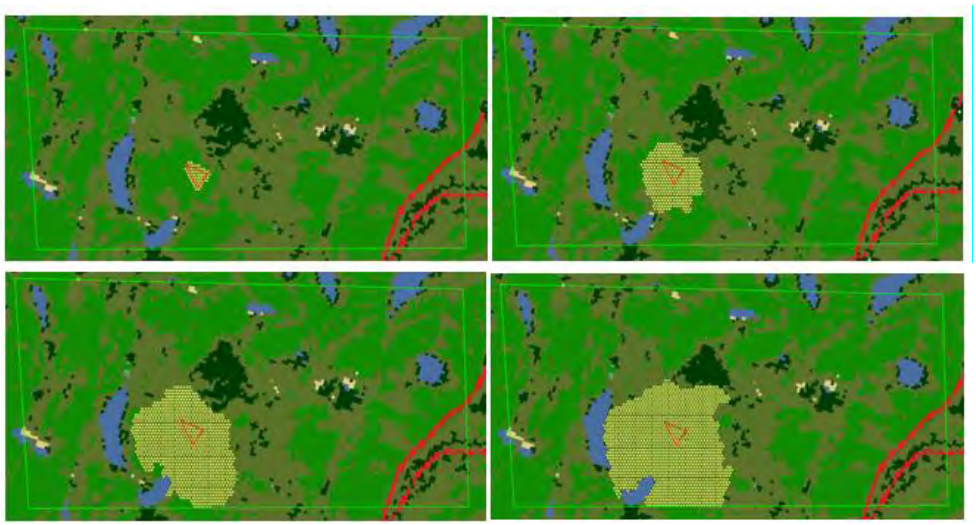

Figure 8: Visualization of simulation output in QGIS.

4 CONCLUSIONS AND FUTURE WORK

This asymmetric Cell-DEVS model takes inputs from publicly available mapping products to determine the elevation and fuel type of each cell, improving the accuracy of the calculation of fire spread when compared to a uniform cell model. The output of the model includes the ability to easily create a time lapse of the burn area overlaid on a landcover map using GIS software. Additional information such as the spread direction and rate are available to researchers. This model can be further modified to add other data such vegetation height, thickness and moisture values, which are increasingly available from satellite imagery. This model demonstrates the suitability of the asymmetric Cell-DEVS formalism implemented in the Cadmium simulator for modelling wildfire spread; allowing practitioners to rapidly create and execute simulations of a multiple areas, under a variety of wildfire conditions in order to provide vital information to decision makers.

REFERENCES

Al-Habashna, A., C. Ruiz-Martin, and G. Wainer. 2019. "Analyzing the Impact of Quantum Size on the Accuracy and Performance of Cell-DEVS Fire Models". In *Proceedings of the Symposium on Theory of Modeling and Simulation (SpringSim-TMS)*, 29th April-2nd May, USA, 1-12.

Albini, F. A., and E. D. Reinhardt. 1995. "Modeling Ignition and Burning Rate of Large Woody Natural Fuels". *International Journal of Wildland Fire* 5(2):81–91.

Andrews, P. L. 2007. "BehavePlus Fire Modeling System: Past, Present, and Future". US Forest Service, Rocky Mountain Research Station, Missoula, Montana.

Andrews, P. L. (2018). *The Rothermel Surface Fire Spread Model and Associated Developments: A Comprehensive Explanation* (Gen. Tech. Rep. RMRS-GTR-371). U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station.

Arroyo, L. A., C. Pascual and J.A. Manzanera. 2008. Fire Models and Methods to Map Fuel Types: The Role of Remote Sensing. *Forest Ecology and Management, 256*(6), 1239–1252.

Balbi, J. H., P. A. Santoni, and J. L. Dupuy. 1999. Dynamic Modelling of Fire Spread Across a Fuel Bed. *International Journal of Wildland Fire* 9:275-84.

Belloli, L., D. Vicino, C. Ruiz-Martín, and G. A. Wainer. 2019. Building DEVS models with the Cadmium tool. In *2019 Winter Simulation Conference (WSC)*, 45–59 https://doi.org/10.1109/WSC40007.2019.9004720.

Cardenas, R., and G.A. Wainer. (2022). "Asymmetric Cell-DEVS Models with the Cadmium Simulator." *Simulation Modelling Practice and Theory 121, Article 102649*.

Cruz, M. G., J. S. Gould, M. E. Alexander, A. L. Sullivan, W. L. McCaw, and S. Matthews. 2015. "Empirical-Based Models for Predicting Head-Fire Rate of Spread in Australian Fuel Types". *International Journal of Wildland Fire* 24(7):849–863.

Dahl, N., H. Xue, X. Hu, and M. Xue. 2015. "Coupled Fire–Atmosphere Modeling of Wildland Fire Spread Using DEVS-FIRE and ARPS". *Natural Hazards* 77(2):1013–1035.

Drury, S. A. 2019. "Observed Versus Predicted Fire Behavior in an Alaskan Black Spruce Forest Ecosystem: An Experimental Fire Case Study". *Fire Ecology* 15(1):35.

Fernandes, P. M. 2009. "Examining Fuel Treatment Longevity Through Experimental and Simulated Surface Fire Behaviour: A Maritime Pine Case Study". *Canadian Journal of Forest Research* 39(12):2529–2535.

Forestry Canada Fire Danger Group. 1992. Development and Structure of the Canadian Forest Fire Behavior Prediction System [Information Report ST-X-3]. Forestry Canada, Science and Sustainable Development Directorate.

Freire, J. G., and C. C. DaCamara. 2019. "Using Cellular Automata to Simulate Wildfire Propagation and to Assist in Fire Management". *Natural Hazards and Earth System Sciences* 19(1):169–179.

Heinsch, F. A. and P.L. Andrews. 2003. *Behaveplus Fire Modelling System: Design and Features* [RMRS-GTR-249]. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station.

Ghisu, T., B. Arca, G. Pellizzaro, and P. Duce. 2015. "An Optimal Cellular Automata Algorithm for Simulating Wildfire Spread". *Environmental Modelling & Software* 71:1–14.

Glitzenstein, J. S., D. R. Streng, G. L. Achtemeier, L. P. Naeher, and D. D. Wade. 2006. "Fuels and Fire Behavior in Chipped and Unchipped Plots: Implications for Land Management Near the Wildland/Urban Interface". *Forest Ecology and Management* 236:18–29.

Grabner, K. G., J. P. Dwyer, and B. E. Cutter. 1997. "Validation of BEHAVE Fire Behavior Predictions in Oak Savannas Using Five Fuel Models". In *Proceedings of the Eleventh Central Hardwood Conference*, Columbia, Missouri, 202–215.

Guo, S., and X. Hu. 2010. "Profile-Based Partition for Parallel Simulation of DEVS-FIRE". *Proceedings of the 2010 Spring Simulation Multiconference*, April 11th–15th, Orlando, Florida, 155–161.

Hu, X., M. Yan, T. Derado, W. Zhao, B. Zeigler, D. Kim, and C. Seo. 2024. "WIP: Towards Cloud-based Wildland Fire Simulation Service". *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, July, San Francisco, California, 20–24.

Little, K., N. Kettridge, C. M. Belcher, L. J. Graham, C. R. Stoof, K. Ivison et al. 2024. "Cross-Landscape Fuel Moisture Differences Impact Simulated Fire Behaviour". *International Journal of Wildland Fire* 33(9):WF24019.

MacLeod, M., R. Chreyh, and G. Wainer. 2006. Improved Cell-DEVS Models for Fire Spreading Analysis. In *Cellular Automata ACRI 2006. Lecture Notes in Computer Science, Vol 4173*, 472–481. Berlin, Heidelberg: Springer.

Mandel, J., J. D. Beezley, and A. K. Kochanski. 2011. "Coupled Atmosphere-Wildland Fire Modeling with WRF 3.3 and SFIRE 2011". *Geoscientific Model Development* 4(3):591–610.

McArthur, A. G. 1962. "Control Burning in Eucalypt Forest". Leaflet No. 80, Commonwealth of Australia Forestry and Timber Bureau, Canberra, ACT.

McGranahan, D. A., D. M. Engle, S. D. Fuhlendorf, J. R. Miller, and D. M. Debinski. 2012. "An Invasive Cool-Season Grass Complicates Prescribed Fire Management in a Native Warm-Season Grassland". *Natural Areas Journal* 32(2):208–214.

Mitchell, R. Rasterio. https://github.com/rasterio/rasterio. Accessed 30th March 2025.

Muzy, A., E. Innocenti, A. Aiello, J.-F. Santucci, and G. Wainer. 2005. "Specification of Discrete Event Models for Fire Spreading". *Simulation* 81(2):103–117.

Ning, J., H. Liu, W. Yu, J. Deng, L. Sun, G. Yang et al. 2024. "Comparison of Different Models to Simulate Forest Fire Spread: A Case Study". *Forests* 15(3):563.

Ntaimo, L., B. P. Zeigler, M. J. Vasconcelos, and B. Khargharia. 2004. "Forest Fire Spread and Suppression in DEVS". *Simulation* 80(10):479–500.

Ntaimo, L., X. Hu, and Y. Sun. 2008. "DEVS-FIRE: Towards an Integrated Simulation Environment for Surface Wildfire Spread and Containment". *Simulation* 84(4):137–155.

QGIS Development Team. 2021. QGIS Training Manual, Version 3.40. https://docs.qgis.org/3.40/en/docs/training_manual/index.html

RMRS Missoula Fire Sciences Lab. Behave: A New Implementation of the Extended Rothermel Model. https://github.com/firelab/behave. Accessed 30th March 2025.

Rothermel, R. C. 1972. "A Mathematical Model for Predicting Fire Spread in Wildland Fuels". Research Paper INT-115, U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Scott, J. H. 2012. *Introduction to wildfire behavior modeling*. National Interagency Fuels, Fire, & Vegetation Technology Transfer.

Scott, J. H., and R.E. Burgan. 2005. Standard Fire Behavior Fuel Models: A Comprehensive Set For Use With Rothermel's Surface Fire Spread Model. *(Gen. Tech. Rep. RMRS-GTR-153). U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station.*

Sousa, F.A., R.J.N. dos Reis, and J.C.F. Pereira. 2012. "Simulation of Surface Fire Fronts Using fireLib and GPUs". *Environmental Modelling & Software* 38:167–177.

Trucchia, A., M. D'Andrea, F. Baghino, P. Fiorucci, L. Ferraris, D. Negro, et al. 2020. "PROPAGATOR: An Operational Cellular-Automata Based Wildfire Simulator". *Fire* 3(3):26.

Wainer G., 2009. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. 1st ed. CRC Press.

Xu, J. 1994. "Simulating the Spread of Wildfires Using a Geographic Information System and Remote Sensing". Ph.D. dissertation, Rutgers University, New Brunswick, New Jersey.

Xue, H., X. Hu, N. Dahl, and M. Xue. 2012. "Post-Frontal Combustion Heat Modeling in DEVS-FIRE for Coupled Atmosphere-Fire Simulation". *Procedia Computer Science* 9:302–311.

Yassemi, S., S. Dragićević, and M. Schmidt. 2008. "Design and Implementation of an Integrated GIS-Based Cellular Automata Model to Characterize Forest Fire Behaviour". *Ecological Modelling* 210(1–2):71–84.

Wolfram, S. 1994. *Cellular Automata and Complexity: Collected Papers*. Reading, Massachusetts: Addison-Wesley.

AUTHOR BIOGRAPHIES

MARK MURPHY is an M.A.Sc. student at Carleton University in Ottawa, Canada. He has a bachelor's degree in mechanical engineering from the Royal Military College of Canada. His research interests include modelling and simulation of aerospace and defence operations. His email address is markmurphy3@cmail.carleton.ca.

JAAN SOULIER is an undergraduate student at Carleton University. His email address is jaansoulier@cmail.carleton.ca.

ALEC TRATNIK is an undergraduate student at Carleton University. His email address is alectratnik@cmail.carleton.ca.

GABRIEL WAINER is a professor at the department of Systems and Computer Engineering at Carleton University. He received is M.Sc. (1993) from the University of Buenos Aires, Argentina, and his Ph.D (1998, highest honors) from UBA/ Université Aix-Marseille-III, France. He is a fellow of SCS. His email address is gwainer@sce.carleton.ca.