Modeling wildfire using evolutionary cellular automata

Maxfield E. Green
University of Vermont
Burlington, Vermont
Maxfield.Green@uvm.edu

Todd F. DeLuca
University of Vermont
Burlington, Vermont
todd.deluca@uvm.edu

Karl WD. Kaiser
University of Vermont
Burlington, Vermont
kwkaiser@uvm.edu

ABSTRACT

With the increased size and frequency of wildfire events worldwide, accurate real-time prediction of evolving wildfire fronts is a crucial component of firefighting efforts and forest management practices. We propose a cellular automaton (CA) that simulates the spread of wildfire. We embed the CA inside of a genetic program (GP) that learns the state transition rules from spatially registered synthetic wildfire data. We demonstrate this model's predictive abilities by testing it on unseen synthetically generated landscapes. We compare the performance of a genetic program (GP) based on a set of primitive operators and restricted expression length to null and logistic models.We find that the GP is able to closely replicate the spreading behavior driven by a balanced logistic model. Our method is a potential alternative to current benchmark physics-based models.

CCS CONCEPTS

• **Applied computing** → **Environmental sciences**; **Physical sciences and engineering**; • **Computing methodologies** → **Agent / discrete models**.

KEYWORDS

Wildfire simulation, Cellular Automata, Genetic Programming

ACM Reference Format:
Maxfield E. Green, Todd F. DeLuca, and Karl WD. Kaiser. 2020. Modeling wildfire using evolutionary cellular automata. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3389836

1 INTRODUCTION

In addition to their natural role in ecosystem dynamics [11], wildfires can morph into natural disasters that threaten human lives, property and ecosystems. Each year, between 4×10^6 and 8×10^6 acres of land are damaged by wildfires, causing USD $5.1 billion cost in infrastructural damage repair over the past 10 years [32]. For example, bushfires in southeastern Australia during 2019 and 2020 have burned $\approx 5 \times 10^6$ ha, destroyed over 2000 homes and killed at least 24 people. Effective wildfire modeling can help inform wildfire response decision making and forest management policy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '20, July 8–12, 2020, Cancún, Mexico
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3389836

We propose FireGen, an evolutionary stochastic cellular automaton model that predicts forest fire growth. Given information about the local ecology, topology and weather systems, a set of probability distributions are learned from time series of birds eye view images of historic wildfires. Given a current fire burning in a known ecosystem, ecological and atmospheric data can be used to predict its burn path. We produce test accuracies up to 74% predicting fire paths based on unseen synthetic data sets. The performance of FireGen is compared to baseline genetic algorithms.

1.1 Wildfire Modeling

Building predictive models to aid in wildfire preparation and containment efforts is an important but difficult task. The coupling of the atmospheric systems and the spreading of fire contribute to the difficulty of modeling the wildfires. Current cutting-edge wild fire models are comprised of sets of coupled nonlinear partial differential equations that estimate the physical processes of ignition and combustion such as reaction diffusion, advection and convection [16, 17, 29]. The US forest service currently uses an ensemble forecasting model implemented by by Finney et al [1, 7]. The model runs many iterations of the FARSITE fire model altering initial conditions and weather predictions based on historical data [6]. These models are computationally costly and cannot be performed using computational resources typically available to forest service departments [1]. Additionally, they can be numerically unstable which can lead to false predictions informing disaster reaction decisions [12, 13]. Assumptions about physical laws are non uniform across different models and thus there is not a universal fire spreading theory, but rather many competing ones. These models are fully deterministic and will yield the same results given the same initial condition. Calculating relative uncertainty in a given prediction can be difficult and require ensemble forecasting.

1.2 Modeling spatial spreading process with machine learning

To reduce computation time, increase accuracy and leverage the advances in satellite imagery, recent work has modeled wildfire dynamics with machine learning and evolutionary strategies. This area has seen great success with increased accuracy of perimeter prediction from historic fires [33], [26]. Crowley et al [8] applied a set of reinforcement learning algorithms to learn spreading policies from satellite images within an agent based model.

Radke et al proposed a deep neural network algorithm titled FireCast that predicts 24 hour wildfire perimeter evolution based on Satellite images and local historic weather [26]. FireCast achieves a 20% higher average accuracy compared to the Farsite model [6] used in current practice.

Spatial spreading processes are commonly simulated using cellular autamaton (CA) [14],[10],[33] and agent based models ABMS[28],[9]. ABMs can been used to simulate complex systems by prescribing rule sets to independent agents. In the case of a wildfire model, each cell on fire represents an agent that can spread across the landscape and ignite neighboring cells based on a probability distribution that considers information about the current neighborhood. By evolving the function that governs agent behavior, agent based models can be used to predict system level spreading based on ground truth data. System level behavior is predicted by fine tuning agent decision functions. For example, Zhong et al [34] modeled evacuation crowd dynamics by evolving the agent rule set. This work aimed to predict the decision making process of an individual in an emergency evacuation of a building. Agents choose which exit to leave a building from based on distance, probable safety and volume of other agents headed that way. An optimal rule set will balance each of those variables to optimize the likelihood that all agents are able to leave the building safely. Fitting a symbolic regression to simulation results using an evolving rule-set exposes a population probability distributions that optimally weigh the considered variables. A number of fields have used this method to build realistic simulations used for further system prediction [4], [19], [18].

We propose a CA that trains a series of genetic programs to replicate seen and unseen wildfire simulations. We first introduce the mechanics of the CA, then give an overview of the evolutionary process and experimental design. We show that the underlying spreading behavior can be learned and replicated by a genetic program based on synthetic environmental features.

2 METHODS

We fit a symbolic regression to data generated by wildfire simulations. A cell in the landscape grid can take on a number of state, including a fire state. Changes in states can spread discretely across the landscape according to a function of a site local Von Neumann neighborhood[30]. We propose a naive spreading function and examine how well the regression can reproduce the spread patterns generated. We compare different genetic programs embedded in a CA by calculating how well they reproduce synthetic burns. We will first describe the CA that simulates the spread of fire and the generation of the synthetic data, then we will discuss the different evolutionary algorithms that attempt to learn the rules that govern the spread of fire.

2.1 Cellular Automaton

CAs were initially proposed by Von Neumann and are used to model spreading dynamics in discrete time and space [21]. CAs are well suited for simulating spreading on a grid. Each site on the grid has attributes that describe its unique state. The behavior of interest spreads across the grid when cells adapt according to their neighbors. In modeling a wild fire, we are interested in the relations between ignition probability and a number of local environmental factors, such as wind speed and direction, temperature, and relative humidity. Over time fire spreads from one cell to another based on a probability distribution that treats these factors as (learn-able) parameters.

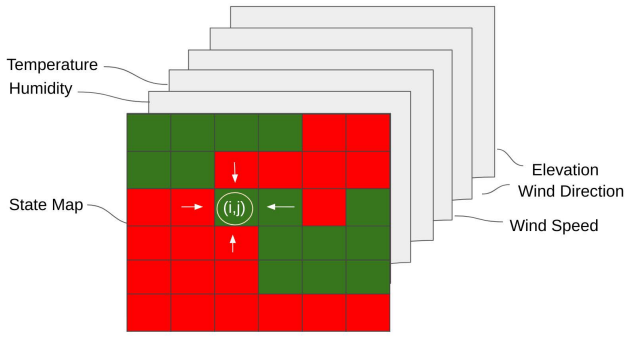


Figure 1: At each time-step of the simulation, all cells that are adjacent to a cell on fire are considered for ignition. The probability of ignition is determined by passing features from the cites Von Neumann neighborhood into an ignition probability distribution. Features are calculated from the 6 layers present in the visualized data structure, temperature, humidity, state, wind direction, wind speed and elevation.

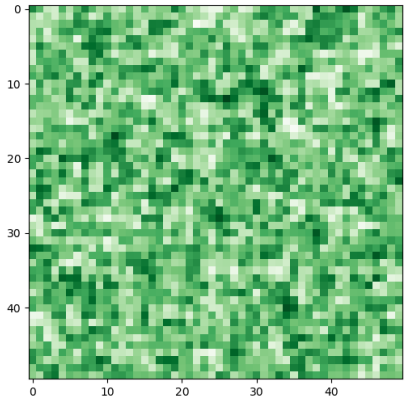
As we show in Fig. 1, each position on the landscape grid has six attributes: elevation, wind direction, wind speed, temperature, humidity, and burn state. The burn state attribute represents the state of the cell, in this case either on fire or not. The Von Neumann neighborhood is defined as the four orthogonal cells as indicated by white arrows in Fig. 1. Spreading behaviors on a grid surface are often modelled using this type of neighborhood [30]. At each time-step, the attributes of the neighborhood of each cell that borders the fire front contribute to the probability that the cell will catch fire.

In traditional CA models, the probability distribution that determines if a cell will adapt the behavior of its neighbor is static and prescribed by the CA modeler. However, in this work, we evolve the spreading function through symbolic regression. In each simulation, the CA runs for t time-steps and each cell updates its state based on its neighbors' attributes. At the end of a time-step, the perimeter of the fire expands probabilistically. At the end of the simulation, we retain an array of the coordinates of ignited cells.

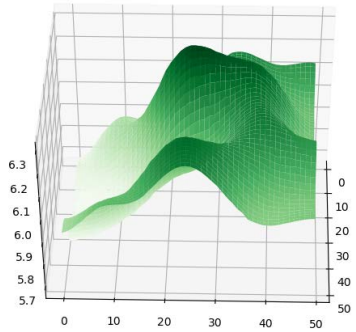
2.1.1 Synthetic data generation. We simulate fires that spread over landscapes made of the six layers described in 1. The state layer is a binary matrix indicating whether a site is on fire or not. To build all other layers we generate matrices of smoothed-random floating points by implementing the Perlin noise algorithm [22] [23]. This iterative technique allows for the user to control how smooth or rough the generated spatial distribution is.

Fig. 2a - Fig. 2b shows the evolution of a Gaussian sample from random noise to a smoothed landscape. Over many iterations, the random sample begins to resemble a realistic smooth landscape, as seen in Fig. 2, which is the result of 100 iterations of this procedure.

We generated each layer of a landscape using Perlin noise due to its abilities to produce natural gradients and its longstanding use in computer generated images [23, 24]. We generated a single layer for each attribute considered in the model (topography, wind speed



(a) Random noise, t_0



(b) Perlin Noise, t_{24}

Figure 2: Random values are sampled from Gaussian distributions and embedded over a matrix. We interpolate slopes between neighboring sites within the matrix, and their neighbors to describe smooth gradients over the sampled values.We use the quintic fade function $6t^5 - 15t^4 + 10t^3$ to interpolate smooth curves between all sites and those within their neighborhood

and direction, etc). We define a binary state layer that changes over the course of a simulation. The binary state layer encodes the state of every cell at a given time as either on fire (1) or not (0). These matrices are then stacked together to form a landscape in which a fire can be simulated on.

2.1.2 Feature engineering. Using the six attributes layers – elevation (Z), wind direction (wd), wind speed (ws), humidity, and burn states (S), we generate a set of four features that describe how the neighborhood affects the ignition likelihood of a central cell. In the case of the temperature and humidity, we take the average over the neighborhood, producing features θ, ϕ, respectively. Since fire is more likely to spread uphill then down, the elevation feature, γ, weighs fire that is downhill from the central cell more than fire that is uphill from the central cell. Fire moves uphill more quickly as reported by the Verisk Wildfire Analysis group, [32]. Thus, we have

$$\gamma = \sum_i I(S_i = 1) e^{-\Delta Z_i} \tag{1}$$

$$\Delta Z_i = Z_i - Z_0, \tag{2}$$

where Z_i is the elevation of the i-th neighbor or the central cell, Z_0 is the elevation of the central cell, and $I(S_i = 1)$ is an indicator variable that is 1 when the i-th neighbor is on fire and 0 otherwise.

Akin to elevation, we define a wind feature that reflects the fact that fire spreads downwind more readily than upwind. The wind feature, ω weighs fire that is upwind more than fire that is downwind. The wind feature is given in equations `Eq.3`, `Eq.4`, where ws represents wind speed, wd represents wind direction with i and j represent the spatial coordinates of the cell on the grid.

$$\omega = \sum_i I(S_i = 1) e^{w_i} \tag{3}$$

$$w_i = ws_i * (cf_i * \cos(wd_i) + sf_i * \sin(wd_i)) \tag{4}$$

where cf_i is a horizontal factor and sf_i is a vertical factor, corresponding to cosine and sine evaluation. Both factors are based on the relative position of the i-th neighbor to the central cell and are used to include the component of the wind that is blowing toward (or away from) the center cell. For example, if the neighbor is north of the central cell, then $cf_i = 0$ and $sf_i = -1$.

These features then become inputs to a probability distribution that determines if a given cell will catch fire based on its neighborhood.

2.1.3 Spread probability distribution. To generate the burn history included in the synthetic data sets, a 5 parameter fixed balanced logistic function is used, See `Eq.5`. Feature vector $[\omega,\gamma,\theta,\phi]$ is given by $\vec{F}$.

$$\text{Logistic Model: } \text{logit}[p(\vec{F})] = \beta + 0.8\omega + 6\gamma + 0.2\theta - 0.2\phi \tag{5}$$

This probability distribution is sampled during fire simulations to build synthetic burn perimeters, as displayed in 3. Once this data was generated, our focus is to see how well the genetic program can evolve a set of functions that reproduces burn patterns. We compare three different models: a null constant model, a logistic model and an unrestricted algebraic model. The null model is composed of a single tunable bias parameter β_0 that is fit to the data. The model, thus, determines only the rate of fire spread, regardless of the neighborhood. Additionally, we consider a 5 parameter logistic model whose β parameters are fit to synthetic data. The output of these functions are probabilities that a given site will catch fire.

$$\text{Null Model: } p(F) = \beta_0 \tag{6}$$

$$\text{Logistic Model: } \text{logit}[p(\vec{F})] = \beta_0 + \beta_1\omega + \beta_2\gamma + \beta_3\theta + \beta_4\phi \tag{7}$$

The unrestricted algebraic model is free to take any form given the bank of potential operators and terminals. The length of the expression is limited to a tree depth of 17 [15]. We will refer to this model as the genetic program model. Comparing these three models represents a good scope of expected performance. The constant

model is a baseline, while the logistic model serves as an upper bound on the accuracy of the genetic program model, as it already has the same functional form as the underlying spreading function, and must only tune coefficients. All three models are evolved under the same set of hyper-parameters and learning schemes.

2.2 Implementation of Genetic Program

Validity and fitness of expressions are subsequently used to select ideal solutions and discard poor ones using tournament selection, with tournament size 4. The best performing individuals will be further subjected to cross-over and mutation. Evolution was implemented using the python library DEAP [27] according to the basic genetic programming as specified by Koza [15].

An individual in the population represents a candidate probability distribution for fire ignition with fitness determined by how well it can reproduce a known fire event using the cellular automata.

Individuals are represented as syntax trees constrained to nodes of primitive operators and terminals. The operator set contains addition, subtraction, multiplication, protected division, negation, and basic trigonometric functions (sin, cosine). The terminal set is comprised of the features from of any given positions neighborhood (e.g. floating point values denoting that positions attributes: elevation, temperature, humidity, wind speed, and wind direction) as well as ephemeral constants in the range [-10,10]. Additionally, the tree is limited to a depth of 17. We impose this limitation to reduce code bloat and over-fitting, a common problem for genetic programs [5]. After a function is evaluated on a cells neighborhood features, a sample from the standardized normal distribution is drawn. If the output exceeds the sample, then the cell will ignite.

2.3 Evaluation of candidate models

Evaluation of the genetic program is conducted under two primary schemes: by evaluating over initial and final states of multiple landscapes, or by evaluating over each time-step of a simulation on a single landscape. This approach captures the ability of an individual to perform well at two timescales, reducing heterogeneity within the population.

2.3.1 Experiment one: Learning from multiple landscapes. To calculate the fitness of an individual, the individual is used to simulate a set of fires across a set of landscapes. The burn simulation produces a predicted burn data set comprised of final states maps for each landscape. From the resulting data, the average intersection over the union (IoU) of the true and predicted state maps for each landscape is calculated. The IoU is commonly used as a cost function in reinforcement learning and image detection settings [2] [20]. The magnitude of the IoU indicates how well the individual predicted the spread of fire in the allotted time window. To generate the reported experimental results, the GP was trained on 10 landscapes and tested on 10 additional landscapes. Training on multiple landscapes puts evolutionary pressure on solutions being able to generalize to different environments. This approach also prevents the GP from simply learning the Perlin noise distributions that generated the synthetic landscapes. We discuss and report the results of this experiment in Figs. 4,5 and 6.

2.3.2 Experiment two: Learning over single timesteps. Alternatively, we introduce another training scheme that prioritizes how well an individual can train on one time-step to predict the next. One time step is defined as the period in which each cell on the landscape grid is considered for ignition once. In this way, we hope to capture (and subsequently evolve) the behavior of the wildfire *on that one specific landscape* at any given time-step rather than its behavior overall. For example, the evolving model is given the burn state of the first time-step of a ground truth burn for a specific landscape, asked to predict the second, then given a fitness equivalent to the IoU of that prediction with respect to the true burn state of the second time-step. These preliminary fitnesses are found using each time-step in the training data, then averaged to provide an overall measure. The only time-step omitted from this process is the last (as there is no subsequent time-step to provide a basis for calculating IoU). This approach was therefore attempted on a separate set of data than the first experiment, but this data was seeded, generated, and given a ground truth 'burn' using the same methods as the initial / final landscape state method.

3 RESULTS

We first describe the behavior of fixed spreading distributions that are used to generate synthetic burn patterns. We then describe how well the genetic program, constant and logistic models performed under two experiments with the goal of reproducing the burn patterns.

3.1 Behavior of biased spread functions

To design a function used to create realistic spreading behavior, we considered features one at a time, and visually analyzed their effect on spreading. Fig. 3 shows spreading according to three biased models and the result of the final balanced logits model.

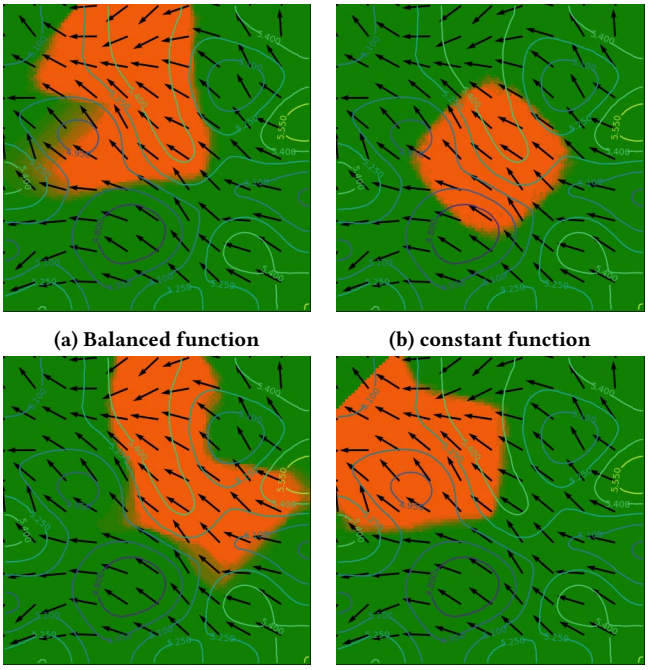
In Fig. 3c, we see the fire spreading uphill along a positive gradient of the landscape. Alternatively, in Fig. 3d, we see the fire following the wind current, moving North West. By balancing the contributions of the different features, we see the spreading behavior in Fig. 3a , with fire spreading in the direction of the wind current with discrimination to the elevation change. The balanced function accounts for all attributes of the landscape and generates more nuanced behavior.

3.2 Model performance of training on multiple landscapes

Simulations were run with a training and test set both of size 10. The evolution lasted 50 generations with populations of size 100. This was done for both the constant and logistic models, followed by the experimental GP model. Each used optimal hyper parameters found from parameter tuning as displayed in Table 1.

In Fig. 4 we see that the constant model shows no change in the distribution of mean fitness of new individuals. Alternatively, the logistic model shows an upward trend; individuals resulting from crossover or mutation thus improved in mean fitness. The genetic programming model demonstrates poor fitness in early function evaluations but very fast improvement.

Fig. 5 shows the mean and standard deviation of the best fitness of each generation, across 16 repetitions.



(a) Balanced function (b) constant function

(c) Elevation-biased function (d) Wind-biased function

Figure 3: Probability distributions that bias burn probability toward specific environmental features. The balanced distribution 3a gives equal weight wind and elevation. While the constant function 3b causes fire to spread stochastically in every direction.

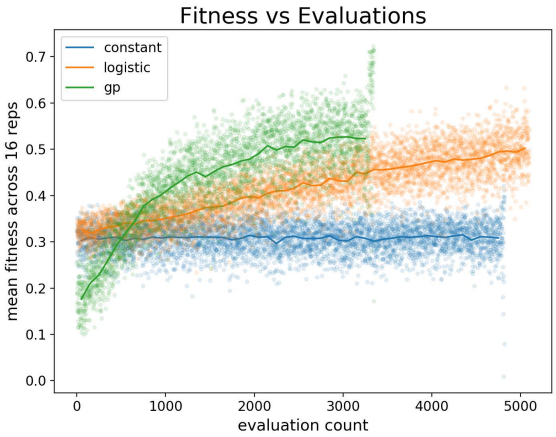


Figure 4: The GP model shows an initial fitness of 0.19, lower then both the constant and Logistic model. Both alternative models have implicitly bounded output while the GP must learn the correct domain over time.

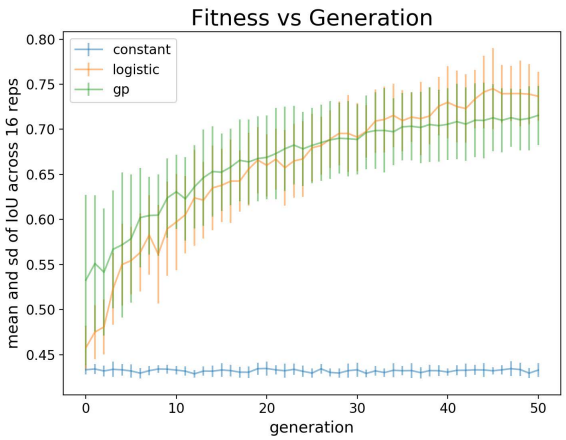


Figure 5: While the GP model initially displays a faster learning rate, the two models converge in performance after 26 generations. The constant model indicates a reference baseline performance of 43%, constant over generations.

We expect in the long run for the logistic model to eventually find the right logits to match the true spreading model. Once this happens, the GP will have a difficulty competing because its solutions are much more complex. We discuss methods to reduce this complexity in the discussion section.

`Fig.6` presents the distribution of maximum fitnesses per repetition from the three models in addition to the ground truth model. The true model is the "balanced logits" model that was used to generate the burn. Due to the stochastic nature of the burn simulation, this model fitness represents optimal fitness. The logistic model comes close to the performance of the true model, as expected. The constant model represents the performance of an extremely simple model. We have separated the training and testing results. While typically, the training results in evolutionary algorithm methods are worse than the validation results, we believe that the variance in the environmental layers me be favoring the validation set.

3.3 Single landscape, multiple time-step evaluations

Another characteristic of a well fitting spreading distribution is the ability to foresee short term changes in the fire front as there are a number of ways that a fire could burn to the final perimeter. We employee the same evolution scheme as the prior experiment but use a different cost function to drive evolution. Individuals are evaluated for fitness after any single time-step of the simulation. We run a simulation according to an individual distribution an evaluate its success at predicting one time step ahead by taking the IoU between the predicted burn set and the true burn set.

We use a population of 100 individuals, each run for 100 generations within each repetition. Again, the results of each repetition were saved, yielding a total of 20 runs from which data could be

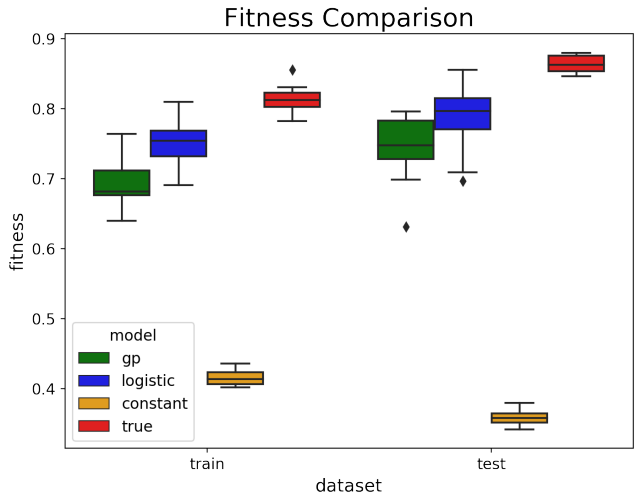


Figure 6: The Logistic and GP models come within %10 of producing the same burn pattern as the underlying spreading function. This indicates that the GP has learned the short term spreading pattern in this environment.

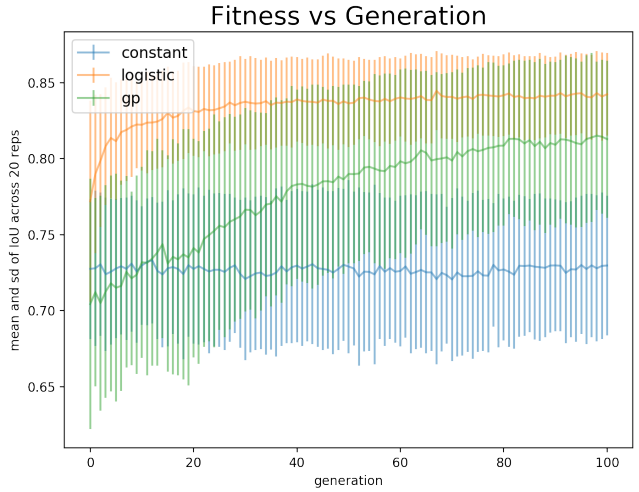


Figure 7: The logistic model learned to 83% accuracy within the first 20 generations. The GP shows a much slower learning rate but approaches the same average fitness. All models display highly variant results as indicated by present error bars.

extracted. Mutation and crossover rates were 0.08 and 0.8 respectively. We first examine the fitness with respect to each generation, see Fig. 7.

The constant model did not change fitness throughout the entirety of the evolutionary run and with each passing evaluation or generation. However, the GP and logistic models demonstrated different behavior than reported in the first experiment, with the logistic method outperforming the GP model in both number of

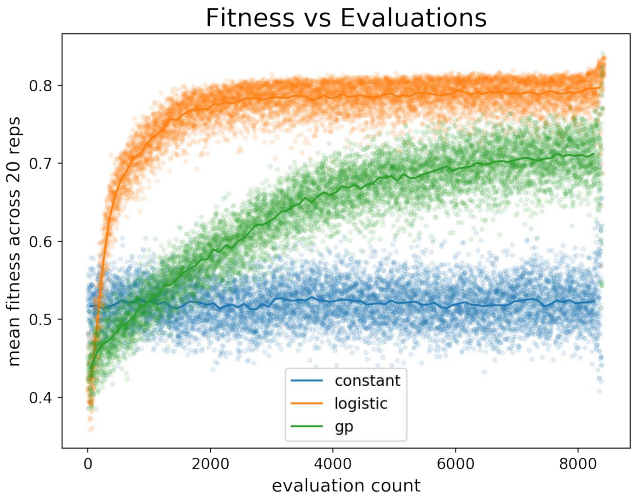


Figure 8: The constant model demonstrated abnormally high performance; even outperforming the GP model in the beginning of the evolutionary run. The GP models grows in fitness over evaluations and approaches the performance of the logistic model.

evaluations to convergence and overall maximum fitness reached. Even considering the variance of each model's data, the logistic method demonstrated a far stronger advantage in the multi time-step experiment.

The constant model demonstrated expected performance, while the logistic and GP models performed similarly to one another. However, the fitness over generations showed high variance in in `Fig. 7` with error bars covering just over a full tenth of the fitness scale (0.1). Furthermore, we examined the overall performance of the models over the course of 20 repetitions in Fig. 9.

While being tested on unseen environments, the constant model performed significantly worse, while the GP and logistic model performed comparably.

4 DISCUSSION

We developed a model that learns the spreading behaviors of synthetic wildfires based on environmental, atmospheric data coupled with historic fire burn perimeters. These data-sets can be synthetic or real. We have shown that the macro spreading behaviors can be learned by evolving the spreading function at differing temporal resolutions. We show that the uninitialized population of algebraic expressions can evolve to produce prediction accuracy's comparable to the true underlying spreading function. We will next discuss some of the structural components of the evolution process.

4.1 Parameter tuning

An essential part of optimizing evolutionary algorithms is setting the correct hyper parameters. We choose to consider mutation and crossover for tuning. Using a grid search, crossover and mutation rate were both tuned to optimize final fitness on a held out validation set. A 5 fold cross validation was used. These parameters were

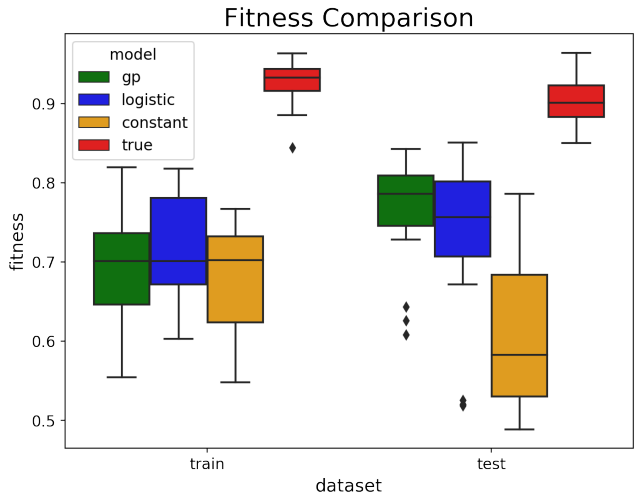


Figure 9: The constant and GP models demonstrated comparable training fitnesses, while the logistic model dominates. The true distribution is used to recreate a 20 fires to represent the true stochastic upper bound on performance.

tuned for the constant and logistic null models under an initial/final landscape fitness function scheme.

The genetic program was tuned by finding the optimal crossover rate sweeping over values [0.5, 0.6, 0.7, 0.8] while holding the mutation rate constant at 0.1. The optimal crossover rate was then used to find the optimal mutation rate sweeping over values [0.1, 0.2, 0.3, 0.4, 0.5] with the same experimental design described in Section 2.3. In future work these hyper-parameters would also be tuned with a grid search.

Table 1: Optimal Hyper-Parameters

Model	Crossover Rate	Mutation Rate
Constant	0.7	0.8
Logistic	0.4	1.0
GP	0.8	0.08

We note that the optimal mutation rate for the Logistic Model is 1. This indicated that the Logistic model is primarily evolving from selection and mutation.

Additionally in the presented result sets, the constant and logistic model often held initial fitnesses significantly higher then the the GP model. We determined that this was due to the GP needing to learn the optimal distributions of outputs to become a true probability distribution. Initial distributions can feasibly contain negative numbers resulting in no spreading. Alternatively, the logistic model will implicitly produce a distribution bounded in [0, 1].

4.2 Optimal Function Forms

While the accuracy distributions of the fittest indicated that the GP can learn a function that will reproduce the spreading patterns, we are also interested in the functional form of the solutions and how close they are to the balanced logistic function. We track the mean and max length of the fittest individuals over 20 reps for 100 generations. The results are displayed in Fig. 10

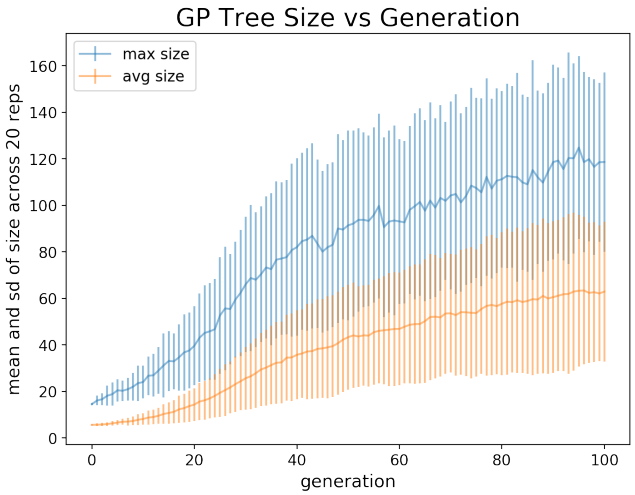


Figure 10: Over time, the size distribution of solution increases. This type of trend can indicate code bloat. However, there is a reduction in size acceleration after the 40 generations.

We note that as the population evolves, the individuals grow larger in length. Code-bloat is a problem common to genetic programs and can lead to over fitting and loss of model generalization. To reduce this problem, in future iterations of the project stricter tree depth or node count limitations could be enforced. A fundamental problem present at this resolution of simulation is heterogeneity of solutions. I.e. multiple solutions can potentially generate the same behavior, achieving the same fitness. Heterogeneity makes uncovering any causal relationships very difficult. We present two example expressions sampled from the final evolved population.

$$\text{True Solution} : \text{logit}[p(\vec{F})] = -7 + 6\omega + 0.2\gamma + 0.2\theta + 0.8\phi$$

$$\text{Fitness} = 0.80 : \text{logit}[p(\vec{F})] = -3.88 + 0.43\omega + 5.25\gamma + 0.10\theta - 0.43\phi$$

$$\text{Fitness} = 0.85 : \text{logit}[p(\vec{F})] = -7.52 + 7.15\omega + 0.22\gamma + 0.27\theta + 0.87\phi$$

We see that both evolved solutions produce high fitnesses but use different coefficients. While this may be suitable for pure prediction tasks, we note that this is a drawback to this method. We conjecture that this may have to do with the simplistic nature of the synthetic data-sets. Further, the the full genetic program produces wild results as shown below.

$$\text{True Solution} : \text{logit}[p(\vec{F})] = -7 + 6\omega + 0.2\gamma + 0.2\theta + 0.8\phi$$

$$\text{Fitness} = 0.71 : (-0.32 + \phi) * \phi$$

$$\text{Fitness} = 0.79 : \frac{\omega^4 * \phi}{\gamma} - 9.76 * \theta$$

These solutions do not closely resemble the true solution, despite having high fitness values. In future iterations, we would like to

constrain the complexity of the solution as a second objective to fitness.

4.3 Limitations and Future Steps

As is the case with many models, there are many assumptions that are held in this model that could be relaxed with additional environmental layers. Most importantly, we have considered fuel sources to be homogeneous and all landscapes are comprised of tree fuel beds. Of course in reality, there are complex distributions of fuel types and this can have a huge effect on fire spreading behavior. Additionally, we assume uniform tree height, which has also been shown to be an important factor in heat transfer and material ignition [25]. We also assume that fire can only spread between neighboring cells on the grid; however, embers can spread to disconnected patches of vegetation starting "spot fires". While these assumptions are clear limitations to the applicability of the model, adding these features to a future model is highly feasible and would not introduce a noticeable increase of complexity.

The obvious next step of this project is further optimization, then validation of the method using real wildfire data. There are several suitable datasets that are available to validate this method, including the 2011 Richardson Wildfire and 2016 Fort McMurray wildfire, both of which took place in Northern Alberta. These data sets are openly available through the NASA's EarthExplorer Data Portal [31]. The accompanying weather data is available through the Canadian Weather service [3]. The results of this experiment could then be directly compared to the recent work by [8]. These two fires serve as a perfect train and test set as they took place in a very similar climate at different times.

Additionally, we hope to apply both experiment one and experiment two as a joint multi object fitness function. This way, individuals that can do short and long term prediction are selected for evolution.

5 CONCLUSIONS

We propose a genetic program embedded inside a cellular autamata simulating wildfires in different synthetic landscapes. We found that the genetic program is able to capture the behavior of the wildfire to produce burns on synthetic data sets that are realistic to burns generated by the underlying spreading function. We summarize some of the main takeaways from this work.

- On average, the GP is well suited to recapture the spreading patterns produced by the balanced logistic function. The GP produces average accuracy's within 15% - 30% of the true spreading function for experiments 1 and 2 respectively.
- Macro spreading behaviors can be learned by tuning the spreading function at differing temporal resolutions
- Evolved solutions are subject to code bloat and do not represent the realistic driving rule-set.

While some of the typical problems with black-box prediction are still present in this model, it is exciting to see that synthetic spreading behavior can be predicted with a moderate accuracy.
This research adds to a new avenue for evolutionary methods to learn spreading rules for cellular automaton based simulations. In the future, we would like to validate this method on a data-set of ground truth remote sensing atmospheric and historic fire perimeter images.

ACKNOWLEDGEMENTS

We would like to acknowledge Dave Landay for his part in the origin of this project. We would also like to thank Dr. Margaret Eppstein , Dr. Chris Danforth, Dr. Peter Dodds and the UVM Complex Systems Center for supporting this research throughout. Additionally, computations were performed on the Vermont Advanced Computing Core supported in part by NSF award No. OAC-1827314.

REFERENCES

[1] Patricia Andrews, Mark Finney, and Mark Fischetti. 2007. Predicting wildfires. *Scientific American. August: 47-55.* 2 (2007), 47–55.
[2] Alexandre Bonnet and Moulay A Akhloufi. 2019. *UAV pursuit using reinforcement learning*. Vol. 11021. International Society for Optics and Photonics, 1. 1102109 pages.
[3] Climate Change Canada. 2019. Government of Canada. (Jul 2019). https://www.canada.ca/en/services/environment/weather.html
[4] Mauro Castelli, Leonardo Vanneschi, and Aleš Popovič. 2015. Predicting Burned Areas of Forest Fires: An Artificial Intelligence Approach. *Fire Ecology* 11 (04 2015), 106–118. https://doi.org/10.4996/fireecology.1101106
[5] Vipul K Dabhi and Sanjay Chaudhary. 2012. A survey on techniques of improving generalization ability of genetic programming solutions. (2012).
[6] Mark A Finney. 1998. FARSITE: Fire Area Simulator-model development and evaluation. *Res. Pap. RMRS-RP-4, Revised 2004. Ogden, UT: US Department of Agriculture, Forest Service, Rocky Mountain Research Station.* 4 (1998), 47.
[7] Mark A Finney, Jack D Cohen, Sara S McAllister, and W Matt Jolly. 2013. On the need for a theory of wildland fire spread. *International journal of wildland fire* 22, 1 (2013), 25–36.
[8] Sriram Ganapathi Subramanian and Mark Crowley. 2018. Using Spatial Reinforcement Learning to Build Forest Wildfire Dynamics Models From Satellite Images. *Frontiers in ICT* 5 (2018), 6. https://doi.org/10.3389/fict.2018.00006
[9] Dirk Helbing. 2012. Agent-based modeling. In *Social self-organization*. Springer, 25–70.
[10] Xiaolin Hu, Yi Sun, and Lewis Ntaimo. 2012. DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models. *Simulation* 88, 3 (2012), 259–279.
[11] Richard L Hutto. 2008. The ecological importance of severe wildfires: some like it hot. *Ecological Applications* 18, 8 (2008), 1827–1834.
[12] Suichi Ichikawa and Shinji Yamashita. 2001. Static Load Balancing of Parallel PDE Solver for Distributed Computing Environment. (2001).
[13] Salim Hariri Jingmei Yang, Huoping Chen and Manish Parashar. 2005. Self-Optimization of Large Scale Wildfire Simulations. *ICCS* ICCS 2005 (2005), 615–622.
[14] Ioannis Karafyllidis and Adonios Thanailakis. 1997. A model for predicting forest fire spreading using cellular automata. *Ecological Modelling* 99, 1 (1997), 87–97.
[15] John R Koza. 1997. Genetic programming. (1997).
[16] Rodman Linn, Jon Reisner, J. Coleman, and S. SMITH. 2002. Studying wildfire behavior using FIRETEC. *International Journal of Wildland Fire* 11 (11 2002). https://doi.org/10.1071/WF02007
[17] Jan Mandel, Jonathan D Beezley, and Adam K Kochanski. 2011. Coupled atmosphere-wildland fire modeling with WRF-fire. *arXiv preprint arXiv:1102.1343* (2011).
[18] Steven M Manson. 2005. Agent-based modeling and genetic programming for modeling land change in the Southern Yucatan Peninsular Region of Mexico. *Agriculture, ecosystems & environment* 111, 1-4 (2005), 47–62.
[19] Steven M Manson. 2006. Bounded rationality in agent-based models: experiments with evolutionary programs. *International Journal of Geographical Information Science* 20, 9 (2006), 991–1012.
[20] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. 2016. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2894–2902.
[21] János Neumann, Arthur W Burks, et al. 1966. *Theory of self-reproducing automata*. Vol. 1102024. University of Illinois Press Urbana.
[22] Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.
[23] Ken Perlin. 2002. Improving noise. In *ACM transactions on graphics (TOG)*, Vol. 21. ACM, 1, 681–682.
[24] Kerman Phillip. 2006. *Macromedia Flash 8 @work, Projects and Techniques to Get the Job Done.* Sam's Publishing.

[25] Justin J Podur and David L Martell. 2009. The influence of weather and fuel type on the fuel composition of the area burned by forest fires in Ontario, 1996–2006. *Ecological Applications* 19, 5 (2009), 1246–1252.

[26] David Radke, Anna Hessler, and Dan Ellsworth. 2019. Firecast: leveraging deep learning to predict wildfire spread. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4575–4581.

[27] De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, Christian Gagné, et al. 2012. Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. ACM, Conference on Genetic and Evolutionary Computation, 85–92.

[28] François Rebaudo, Verónica Crespo-Pérez, Jean-François Silvain, and Olivier Dangles. 2011. Agent-based modeling of human-induced spread of invasive species in agricultural landscapes: insights from the potato moth in Ecuador. *Journal of Artificial Societies and Social Simulation* 14, 3 (2011), 7.

[29] O. Séro-Guillaume and J. Margerit. 2002. Modelling forest fires. Part I: a complete set of equations derived by extended irreversible thermodynamics. *International Journal of Heat and Mass Transfer* 45, 8 (2002), 1705–1722.

[30] Tommaso Toffoli and Norman Margolus. 1987. *Cellular automata machines: a new environment for modeling*. MIT press.

[31] Usgs. [n. d.]. Home. ([n. d.]). https://earthexplorer.usgs.gov/

[32] Verisk. [n. d.]. 2019 Verisk Wildfire Risk Analysis: Property Underwriting. ([n. d.]). https://www.verisk.com/insurance/campaigns/location-fireline-state-risk-report/

[33] Zhong Zheng, Wei Huang, Songnian Li, and Yongnian Zeng. 2017. Forest fire spread simulating model using cellular automaton with extreme learning machine. *Ecological Modelling* 348 (2017), 33–43.

[34] Jinghui Zhong, Linbo Luo, Wentong Cai, and Michael Lees. 2014. Automatic rule identification for agent-based crowd models through gene expression programming. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1], 1125–1132.