Automated Dynamic Resource Allocation for Wildfire Suppression

J. Daniel Griffith, Mykel J. Kochenderfer, Robert J. Moss, Velibor V. Mišić, Vishal Gupta, and Dimitris Bertsimas

Wildland fires pose a significant threat to life and infrastructure, and suppression efforts are costly in manpower and resources. Although important advances in situational awareness tools and communication technologies have greatly aided incident commanders in directing suppression efforts, there is still a lack of effective autonomous dynamic decision support systems that provide resource allocation recommendations to help commanders cope with the uncertainty of evolving fire dynamics and time pressures.

Between 2002 and 2012, the U.S. Forest Service and the Department of the Interior spent on average $3.13 billion per year on wildland fire protection [1]. Nearly $1 billion of that funding was devoted solely to fire suppression efforts. Currently, allocating resources for fire suppression is a subjective process, relying in many cases on individual commanders' decisions that are based on insufficient information. An automated system would help commanders cope with situational uncertainty, time pressures, and limited resources by providing actionable recommendations [2].

Tactical wildland fire management is one example of problems involving resource allocation in highly uncertain, dynamic environments. Similar problems include urban search and rescue, flood monitoring and management, and earthquake response. Such problems involve high dimensionality (e.g., large number of locations or resources), uncertain dynamics, many combinations of resource assignments, and the balancing of multiple competing objectives.

Dynamic resource allocation (DRA) problems are special cases of a more general class of problems called Markov decision processes (MDP) (see sidebar titled "Markov Decision Processes" on the following page). MDPs are a mathematical formulation of problems in which the system dynamics are partially random and partially controllable by a decision maker. MDPs are a reasonable model for wildfire suppression as well as for many other autonomous systems problems in defense. For example, MDPs can also be used to model fleet protection, military logistics, mine countermeasures, surveillance missions, battle management, com-

mand and control, and many other defense applications. In fact, some of the earliest work with MDPs was in operations research for the military.

This article explores two general approaches for finding approximate solutions to DRA problems represented as MDPs. The first approach involves a method called Monte Carlo tree search (MCTS), which adaptively samples future trajectories (or time sequences of states) to various sampling depths from the probability distribution specified by the MDP. It is known that MCTS will asymptotically converge to the optimal solution in the limit of samples, but its performance on DRA problems has not, until now, been empirically studied. An alternative is to use what we loosely term mathematical optimization (MO). The essential idea is to approximately model the MDP as a mixed-integer linear optimization problem and use a commercial solver to find either optimal or high-quality, feasible solutions.

Empirical studies involving the wildland fire problem indicate that MCTS handles smaller problems better than MO does because MCTS is attempting to approximate the solution to an exact model, whereas MO is attempting to find the exact solution to an approximate model.

Markov Decision Processes

Markov decision processes (MDP) are a general framework for formulating sequential decision problems [a]. The concept has been around since the 1950s, and it has been applied to a wide variety of important problems. The idea is simple, but the effective application can be very complex. Figure A shows a small MDP with three states. Available from each state is a set of actions. Actions A and B are available from all three states. Depending on the current state and the action taken, the next state is determined probabilistically. For example, if action A is taken from state 2, there is a 60% chance that the next state will be 1 and a 40% chance the next state will be 2. The benefits or rewards of any action are generated when transitions are made. Rewards can be positive, such as +1 and +5 in the example, or they can be negative, such as -10 for making the transition from state 3 to state 2 by action B. The objective in an MDP is to choose actions intelligently to maximize the accumulation of rewards or, equivalently, minimize the accumulation of costs. Here, for example, action A from state 3 shows the highest reward (+5).

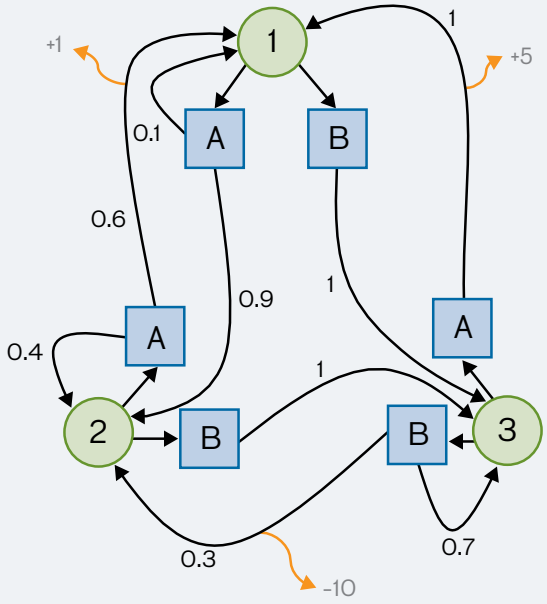

FIGURE A. This simple, three-state system depicts the principal features of a Markov decision process. From each state (green circles), a decision must be made between action A or B. Depending on which action is selected, the system will transition to some new state according to the probabilities shown in the diagram (black numbers). Rewards (orange arrows) are assigned to certain transitions on the basis of the advantages of the subsequent state (e.g., suppressed fires).

Reference

a. R. Bellman, *Dynamic Programming*. Princeton, N.J.: Princeton University Press, 1957.

However, MO tends to perform well with larger state and action spaces when MCTS is not able to adequately sample the relevant spaces. The difficulty with MO is that it outputs only feasible solutions, which may or may not be truly optimal, and it may not find a feasible solution in a reasonable amount of time. In contrast, MCTS is an anytime algorithm and will use whatever processing time is available and, if it gets interrupted, it can still output the best solution found so far in its search.

The contributions of this article are the following:

- The MCTS-based approach for a problem inspired by tactical wildfire management is significant as it represents the first application of MCTS to a DRA problem motivated by a real-world application. This approach combines a number of classical features of MCTS, such as banded upper confidence bounds, with new features, such as double progressive widening (see page 18). Furthermore, we also develop a custom heuristic for the problem to use as a tree policy and rollout policy, and a custom action-generation approach to cope with the size of the state and action spaces.
- An MO formulation approximates the original discrete and stochastic elements of the problem by suitable continuous and deterministic counterparts. Although this approximation is in the same spirit as discussed in other fluid approximation literature in operations research, our particular formulation incorporates elements of a linear dynamical system. We believe these elements may be of independent interest in other DRA problems.
- The MCTS and MO approaches both produce high-quality solutions, generally performing as well as or better than a customized heuristic for this problem. MCTS appears to have a slight edge over the MO approach when the state space of the problem is small. With a fixed computational budget, however, the MO approach begins to outperform the MCTS approach as the problem instance grows, either in state space or action space. Indeed, for very large action spaces, MCTS can

Mixed-Integer Linear Optimization

Mixed-integer linear optimization refers to optimization problems in which (1) all of the constraints and the objective functions are linear functions of the controls and (2) some or all of the controls are constrained to take integer values. As a special case, these controls may further be constrained to be binary, i.e., 0 or 1. Many types of deterministic optimization problems, including the traveling salesman problem (minimizing the total distance or time covered between multiple points) and certain scheduling problems, can be formulated as mixed-integer linear optimizations [a].

There exist computational complexity results suggesting that mixed-integer linear optimization is computationally intractable for large instances from a theoretical point of view. From a practical point of view, however, a number of commercial and open-source software codes solve problems of this type, even for very large instances. Over the last 25 years, a 200-billion-factor speedup in these codes has resulted from algorithmic advances and improved computer hardware. (For example, the notorious traveling salesman problem for an area the size of the United States can now be solved in a few seconds on an iPhone [b].)

Thus, contrary to theoretical complexity results, we consider mixed-integer linear optimization to be practically tractable for many problems and will later provide evidence supporting this opinion with respect to our model.

References

a. W. Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation.* Princeton, N.J.: Princeton University Press, 2012.
b. L. Wolsey, *Integer Programming.* Somerset, N.J.: Wiley Press, 1998.

begin to perform worse than our baseline heuristic. The MO approach by comparison still performs quite well.

Wildland Fire Dynamics

Several methodologies proposed in the literature for simulating fire dynamics differ with regard to the features they capture. In the United States, FARSITE simulation software is frequently used by federal land management agencies to predict fire growth [3]. Both FARSITE and a more recently proposed revision of the model, Prometheus [4], use deterministic simulators to model fire growth. These models are particularly suited for studying the underlying causes of wildland fires and representing the relationships between spread rate, terrain slope, fuel type (e.g., dry tinder, small shrubs, large trees), and wind [4, 5]. Although these models are capable of representing many features of wildland fire dynamics with high fidelity, they neglect the sources of randomness in the spread. Consequently, these models may be unreliable at the time scale needed for tactical management.

Stochastic models (e.g., those in [6–8]) provide an alternative to deterministic models. In general, these approaches divide the terrain into a collection of cells, each of which may be burning. A fire in one cell may spread to nearby ones. Stochastic models often carry a larger computational burden; they must consider multiple possible trajectories for the fire spread from each cell. Nonetheless, uncertainty in the rate and direction of spread is a key challenge in tactical fire management.

Most deterministic and stochastic models used today are based on early fire-spread models developed by Rothermel in the 1970s [9]. The mathematical representation characterizes the fire-spread rate and intensity as a function of surface fuel type and load, terrain slope, wind, and moisture content of the fuel. This representation allows the specific conditions of a wildland fire to inform fire-spread predictions.

Tactical wildland fire managers have access to a number of resources to combat a wildland fire: firefighter crews, heavy equipment such as bulldozers, and aerial drops [10]. Various tactics can be used to apply these resources. One is a direct attack in which the resource works directly on the fire. An example of a direct attack is dropping water or suppressant chemicals on the fire. Indirect attacks generally involve allocating the resource at some offset from the fire and having resources take action that prevents the spread of the fire later on. For example, firefighters may build a fire line or perform a controlled burn to remove surface fuel from the terrain. Some work has been done on modeling the effects of resources on fire spread [11]; however, these models are limited compared to the mathematical models for describing fire spread.

Modeling Fire Dynamics

The MDP to model tactical wildland fire management is inspired by the stochastic model for wildland fire simulation [6]. Several key features of the problem include stochastic spread over time and this spread's dependence on fuel and weather patterns.

An environment where a fire is active or where a fire might start in the future is broken down into segments, such as hillsides, valleys, areas with high tinder content, and areas of increased threat (populated areas). Each segment of the entire landscape is partitioned into a grid of cells. There are two attributes for each cell:

- $B(x)$, a Boolean variable indicating whether the cell x is currently burning, and
- $F(x)$, an integer variable indicating how much fuel is remaining in cell x.

The collection of these attributes over all cells in the grid represents the state of the MDP.

The MDP decisions correspond to assigning a set of suppression resources fighting the wildland fire, $Q(x)$, applied at location x. For each suppression resource, denote the cell to which it is applied as i. It must be assumed that any resource can be assigned to any cell at any time step. If the suppression resource is an aerial water drop from an aircraft, for example, then the assumption must be made that the travel time between cells is small compared to the decision interval in the Markov process.

The fuel in an ignited cell decreases at a constant rate. Without loss of generality, we can rescale the units of fuel in each cell to assume that this rate is one unit. Thus, we model the evolution of fuel in the model by

$$F_{t+1}(x) = \begin{cases} F_t(x) & \text{if } B(x) \cup F_t(x) = 0 \\ F_t(x) - 1 & \text{otherwise.} \end{cases}$$

Notice this evolution is deterministic given $B_t(x)$. The reward for a cell burning is $R(x)$ (always negative) and the

Deterministic versus Probabilistic Models

Our formulation for wildfire dynamics resembles Conway's Game of Life [a], a fully deterministic game played on a (theoretically infinite) two-dimensional grid, but with an important difference. Once the initial configuration is set, the subsequent state for each cell is set deterministically. In the Game of Life, an initial state is set by activating some squares (living cell); this action can be done randomly or by specified patterns. Notice that each cell of the grid has eight neighboring cells—four adjacent orthogonally, four adjacent diagonally. The rules of the game govern three states:

- Survivals. Every living cell with two or three neighboring cells survives for the next generation.
- Deaths. Each living cell with four or more neighbors dies (is removed) from overpopulation. Every cell with one or no neighbor dies from isolation.
- Births. Each empty cell adjacent to exactly three neighbors is a birth cell.

Since each cell has at most eight neighbors, these dynamics parallel the sparse probabilistic $P(x, y)$ matrix later described in the main text. It is important to understand that all births and deaths occur simultaneously. Together they constitute a single generation. Figure A shows a few initial states and their subsequent few steps, some leading to stability (e.g., burning locations fixed until their fuel runs out).

In the realm of wildfire suppression, the prior three rules can be restated as follows:

- Survivals. There is enough fuel in a cell and nearby heat to keep the cell burning.
- Deaths. The fuel in a cell is completely consumed and the fire is extinguished.
- Births. Nearby cells ignite the fire in a new location.

In this deterministic model, fire suppression could be applied to exact locations with perfect foreknowledge of where the fire was going to spread, killing a fire within a very few number of steps. In our model that has random transitions, however, we do not have perfect foreknowledge, and suppression may take many time steps as the fire spreads unpredictably.

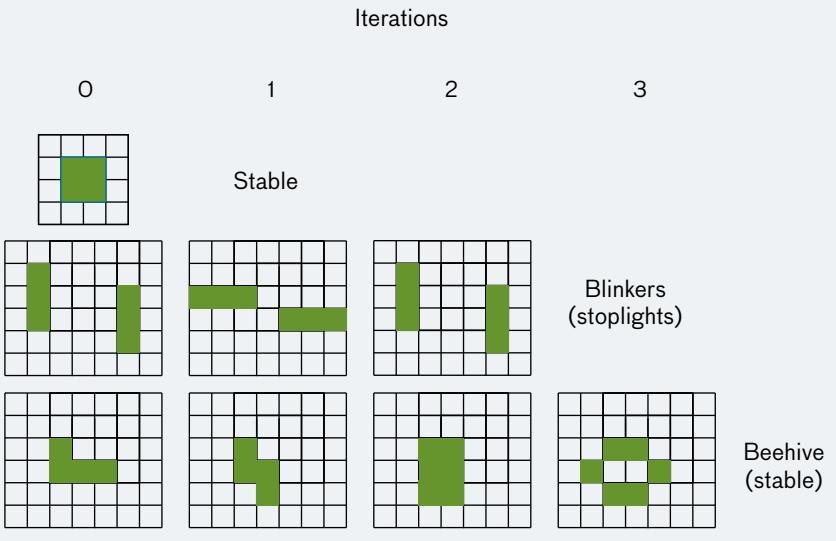

FIGURE A. Several configurations of three and four hot-spot initial states show the effects of deterministic actions on cells. Some configurations achieve stability, others grow forever, others disappear, and still others cycle back and forth through states. Compare this figure to Figure B, in which the current state is partially unknown and the direction of the fire path is unknown. Blinkers and beehive are Conway's names for the bottom two configurations respectively.

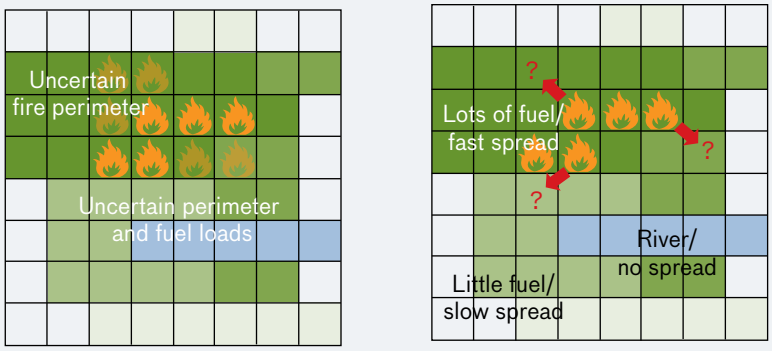

FIGURE B. Conditions near a wildfire might give the fire-suppression coordinator good indications of where a fire might spread, but the situation is probabilistic and uncertain. On the left, the uncertainty is in the current state of the fire; on the right, the direction of the spread of the fire is uncertain.

Reference

a. M. Gardner, "Mathematical Games—The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Scientific American,* vol. 223, 1970, pp. 120–123.

total reward received every step is $\sum_{t,x} B_t(x)R(x)$. The reward across the grid varies to represent a higher cost of a fire in particular areas. For example, we may penalize a fire in a populated area more heavily than one in open terrain. The evolution of $B_t(x)$, however, is stochastic. Figure 1 where $P(x, y)$ is the probability that a fire in cell y causes cell x to burn during the time step and

$$\rho_1 = \begin{cases} 1 - \prod_y (1 - P(x,y)\, B_t(y)) & \text{if } F_t(x) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\rho_2 = \begin{cases} 0 & \text{if } F_t(x) = 0 \\ 1 - \prod_i (1 - Q(x)\, \delta_x(a^{(i)})) & \text{otherwise.} \end{cases}$$

Generally, only the neighbors of x can ignite x, and so we expect $P(x, y)$ to be sparse. (See the sidebar titled "Deterministic versus Probabilistic Models" for a comparison of two distinct methods of fire ignition.) The specification of $P(x, y)$ can capture the tendency of a fire to propagate primarily in one direction because of wind. Later in this article, we discuss how $P(x, y)$ may be calibrated to environmental data like wind direction. $Q(x)$ is the probability that a suppression effort on cell x successfully extinguishes the cell and $\delta_x(a^{(i)})$ is an indicator function, which is 1 when a suppression resource is applied to cell x and 0 for cells in which no suppression effort is applied. The probability of success for multiple suppression attempts on the same cell is assumed to be independent. It must be noted that under these dynamics, cells that have been previously extinguished by a suppression team may later reignite.

Calibrating Fire Dynamics with Environmental Data

In order to model a specific fire, the values for the model parameters can be estimated from environmental data. Specifically, the transition matrix $P(x, y)$ allows a great deal of flexibility in incorporating environmental data (e.g., hills, types and amounts of fuel). This transition matrix can

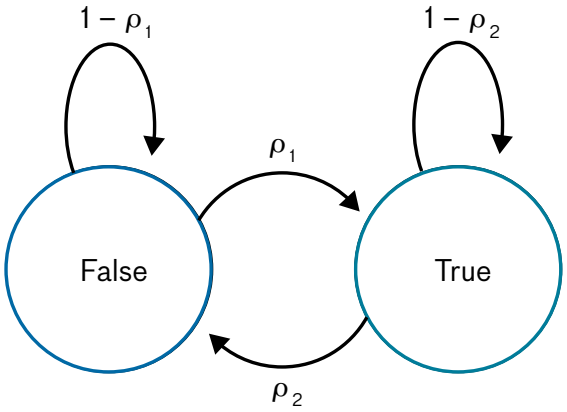

FIGURE 1. The stochastic transition model $B(x)$ can be represented by two states, false (no fire) and true (fire), and by the relative changes from false to true and true to false. ρ_1 represents the probability of a location catching fire, and ρ_2 represents the probability of the fire at a location being extinguished. In general, ρ_1 is larger when neighboring cells are on fire and when certain environmental conditions hold (e.g., dry weather, dry or dead vegetation, upslope terrain). Similarly, ρ_2 is larger when the surface fuel is expelled in the area or when more suppression efforts are applied to an area.

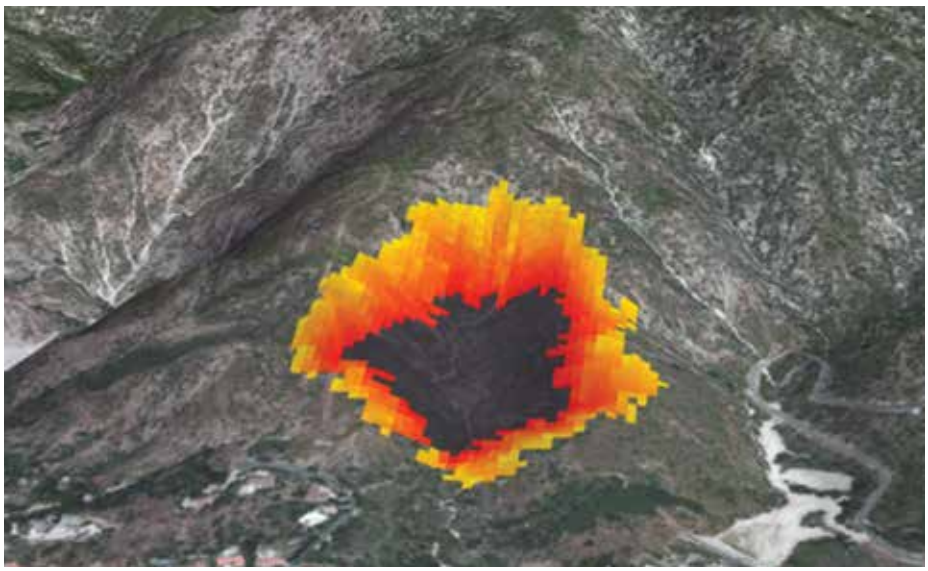

FIGURE 2. This illustration shows a simulated fire with dynamics based on environmental data. The fire is spreading faster uphill (toward the top of the figure) in this simulation. In such a case, although the cost for the homes located downhill to the fire might be significantly higher than that for open woods, suppression efforts should focus on the uphill regions.

be calibrated by leveraging existing deterministic models for wildland fire dynamics [6]. Specifically, we let

$$P(x, y) = 1 - e^{-\lambda \Delta t},$$

where the rate of spread λ is described as a function of

- the location of (and therefore distance between) x and y,
- direction of y to x, relative to the
- direction (and speed) of the wind, and
- a constant λ_b .

The constant λ_b is a terrain-specific base rate of spread, calculated from the Canadian Forest Fire Behavior Prediction System [12]. As described therein, λ_b is a function of the fuel type and assumes zero wind speed. The maximum rate of spread λ_{max} is a function of both the fuel type and the current wind speed. The computed rates of spread, λ_b and λ_{max}, include a slope spread factor that is a function of the terrain elevation angle—a fire will spread faster uphill than downhill. Figure 2 shows our visualization tool's depiction of a simulated wildland fire. In this simulation, the fire spreads faster uphill. The appendix describes in further detail the visualization tool and simulator that we developed.

Monte Carlo Tree Search

One of the most successful sampling-based online approaches for solving MDPs in recent years is MCTS, illustrated in Figure 3 [13]. In contrast with many other online methods, the complexity of MCTS does not grow exponentially with the planning horizon (depth and breadth of the search). In MCTS, a generative model produces samples of the next state and reward given the current state and action. All of the information about the state transitions and rewards is represented by the generative model; the state transition probabilities and expected reward function are not used directly. The generative model can be thought of as a simulator.

The initial algorithm involves running many simulations from the current state while updating an estimate of the state-action value function $Q(s, a)$—that is, the expected reward for taking an action a from a given state s. There are three stages in each simulation:

- A search stage is initiated if the current state in the simulation has already been visited during the execution of the algorithm. During the search stage, the state-action value function is updated for the states and actions visited and tried. The number of times that a state is visited, $N(s)$, is maintained, as is the number of times action is taken from a state $N(s, a)$. During the search, we execute the action that maximizes

$$Q(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}},$$

where c is a parameter that controls the amount of exploration in the search. The square-root term is an exploration bonus that encourages selecting actions that have not been tried as frequently (i.e., new states).

- The expansion stage occurs once a state that has not yet been visited is reached. We iterate over all of the actions available from that state and initialize $N(s, a)$ and $Q(s, a)$ with $N_0(s, a)$ and $Q_0(s, a)$, respectively. The functions N_0 and Q_0 can be based on prior expert knowledge of the problem; if none is available, then both functions can be initialized to 0.
- The rollout (depth search) follows the expansion stage. We simply select actions according to some rollout (or default) policy π_0 until the desired depth (level of action and appropriate result) is reached. Typically, rollout policies are stochastic; thus, the action to execute is sampled. The rollout policy does not have to be close to optimal, but it is a way for an expert to bias the search into areas that are promising. The expected value is returned and is used in the search to update the value for $Q(s, a)$ for the search phase.

Simulations are run until some stopping criteria are met, often simply a fixed number of iterations. The action that maximizes $Q(s, a)$ is executed. Once that action has been executed, the MCTS is run with the new $Q(s, a)$ to select the next action. It is common to carry over the values of $N(s, a)$, $N(s)$, and $Q(s, a)$ computed in the previous step. Figure 3 shows examples of the search tree after different numbers of algorithm iterations.

Double Progressive Widening

Monte Carlo tree search with double progressive widening (MCTS-DPW) is a variation of MCTS that explicitly controls the branching factor of the search tree [14]. This variation is specifically necessary when the action space is continuous or so large that all actions cannot possibly be explored. For example, consider the case in which the number of permissible actions is greater than the number of iterations; the standard MCTS algorithm presented in the last section will never expand the search tree past a depth of one step.

The modified algorithm applies the simulate function used in MCTS-DPW to control the number of actions and the number of next states considered from state s:

1. The first progressive widening controls the number of actions considered from a state. A new action is generated only if the number of actions already explored from a state $N(s, a)$ is below a certain parameter-defined limit. A default strategy for generating new actions is to randomly sample from candidate actions. Once $N(s, a)$ reaches the limiting number, we execute the action that maximizes

$$Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}},$$

the same function as in the standard MCTS.
2. Next, a sample (s', r), the transitioned state and its reward, is selected. New parameters related to state s' limit the number of states transitioned from s. We then continue our search from s', the new state.

Action Generation

As previously mentioned, a default strategy for generating new actions during the search stage of the algorithm involves randomly selecting an action from all candidate actions. Another approach is to randomly select actions that focus on a portion of the action space defined by some heuristic. Eventually both of these strategies will sample the entire action space; however, a downside of these approaches is that action generation does not leverage any information received during previous iterations of the algorithm. Consider MCTS after several iterations. Prior $Q(s, a)$ states and actions could provide some indication of the portions of the action space from which it is promising to sample.

Another approach involves using the estimates of $Q(s, a)$ to bias the sampling procedure through a sampling scheme inspired by genetic algorithm search heuristics [15]. The approach outperforms the default strategies. It involves generating actions using one of three approaches with individual probabilities for each:

- An existing action in the search tree is mutated,
- Two existing actions in the search tree are recombined by joining allocation subsets from each of the actions, or
- A new random action is generated from the default strategy.

When actions mutate or recombine, the existing action (or actions) is selected from $A(s)$ using a method in which the fitness for each action is proportional to $Q(s, a)$.

Rollout Policy

A heuristic for the rollout policy, or action state, in MCTS assigns a reward weight to each cell x,

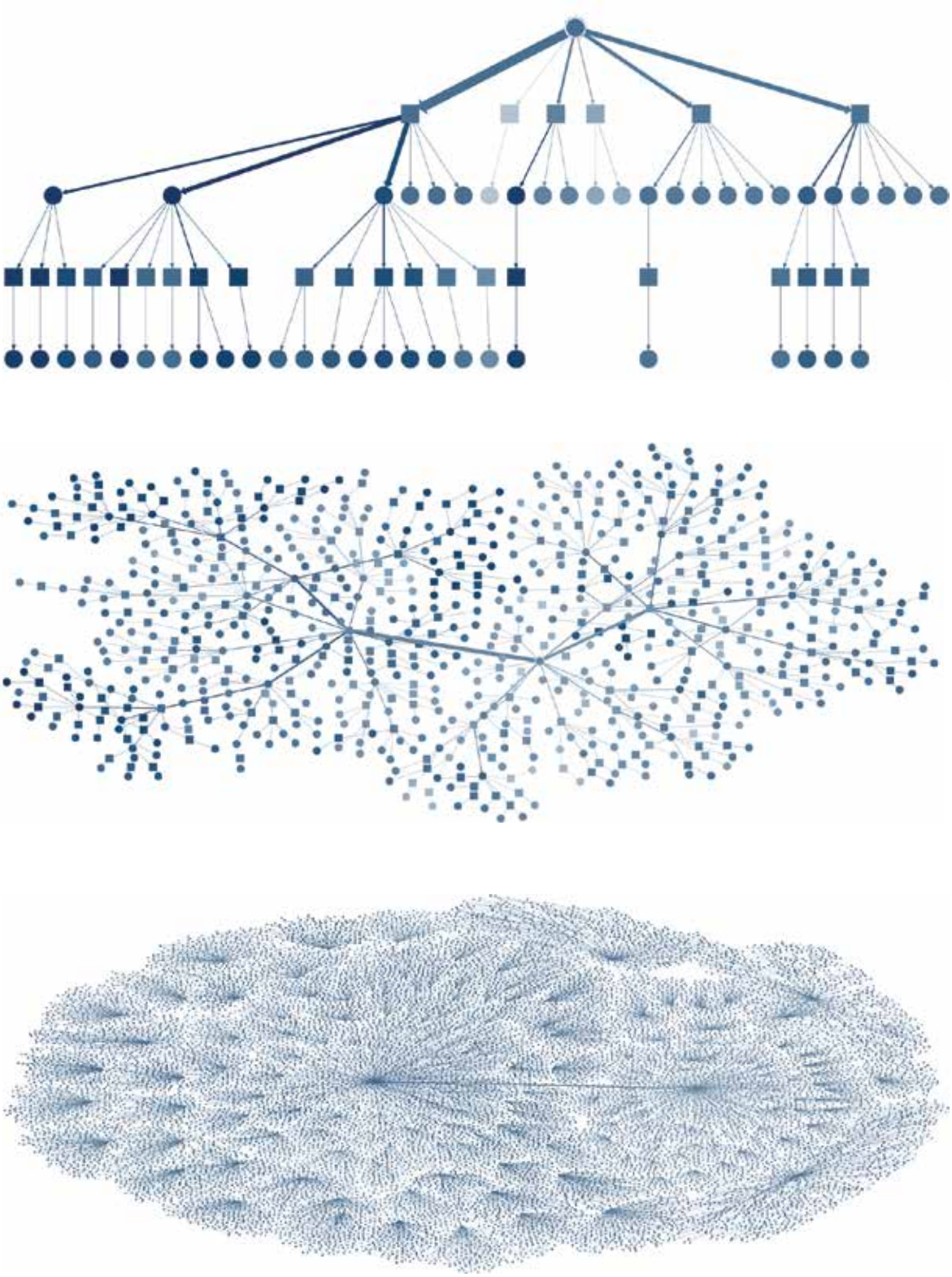

FIGURE 3. Representative examples of MCTS search trees can show additional information beyond simple connections. Here, circles correspond to states, while squares correspond to actions. The thickness of the intersecting lines is proportional to the number of times that the part of the tree has been searched. Darker node colors correspond to higher $Q(s, a)$ estimates.

$$W(x) = \sum_{y} \frac{R(y)}{D(x,y)},$$

where $R(y)$ is the reward for cell y and $D(x, y)$ is the shortest distance between x and y assuming that the distance between adjacent cells is $P(x, y)$. Values for $D(x, y)$ can be determined using a graph analysis algorithm, such as the Floyd-Warshall (FW) algorithm [16]. The weight for each cell is calculated offline because the weight does not depend on how much fuel is left or whether or not a cell is burning.

During MCTS, the rollout policy involves selecting the cells that are burning and assigning resources to the highest-weighted cells. We are also able to generate candidate actions by randomly sampling the corresponding weights. Figure 4 shows example weights assigned to different cells on an eight-by-eight grid with varying reward profiles.

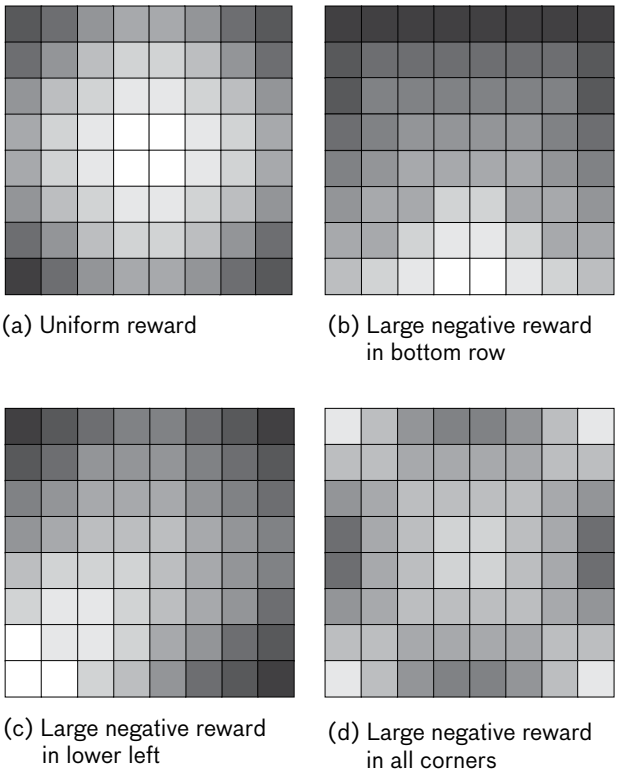

FIGURE 4. Various example heuristic weights were used in the rollout policy. Lighter-shaded cells correspond to higher weights and are generally applied to cells closest to the highest-value cells.

Mathematical Optimization

A deterministic optimization problem approximating the original MDP is applied at each decision step. This approximation is resolved based on the current state of the process, and the first prescribed allocation of resources is selected. These types of optimization schemes are sometimes referred to as model predictive control.

The key feature of the MO formulation is the use of a deterministic, "smoothed" version of the dynamics presented earlier. Rather than modeling the state with a discrete level of fuel and binary state of burning, fuel is modeled as a continuous quantity, as is a new (continuous) intensity level of each cell representing the rate at which fuel is consumed. Other authors in the operations research literature have used similar ideas when motivating various fluid approximations. For example, continuous fluid approximations have been used to study the control of queuing networks [17]; the size of each queue in the network can be modeled continuously through systems of differential equations, and the decision at each time step for each server in the network is the "rate" at which each queue is being served or emptied. Another example comes from revenue management. In a typical revenue management problem, the decision maker needs to sell some fixed inventory of a product and must at each point in time set a price for this product. The price determines the rate of a stochastic demand process, with the goal of maximizing the total revenue at the end of the selling period. The optimal solution of the exact problem is, with few exceptions, extremely difficult to obtain. However, if the dynamics of the problem are relaxed so that the demand is continuous and arrives at a deterministic rate that varies with the price, one can obtain useful bounds and near-optimal pricing policies for the exact stochastic problem [18].

Smoothing the dynamics allows for two important simplifications. First, the probabilistic dynamics described earlier can be replaced with simpler, deterministic dynamics governing the evolution of the intensity of the fire. Second, and most importantly, it is no longer necessary to consider the entire exponentially large state space of the MDP but rather only its evolution along the one path defined by the deterministic dynamics.

Optimization Model

Let $A_t(x, i)$ be a binary variable that is 1 if suppression resource i is assigned to cell x at time t, and 0 otherwise;

A_t is the main decision variable of the problem. Recall that $F_t(x)$ denotes the amount of fuel available at the start of period t in cell x. Furthermore, let $I_t(x)$ represent the intensity of the fire in cell x at time t. Intensity is a continuous decision variable that will be determined by the optimization algorithm. Unlike in the original MDP formulation, it is no longer possible to rescale the parameters so that exactly one unit of fuel is consumed per period.

The objective of this process is the sum of the intensities over all of the time periods and over all cells, weighted by the importance factor of each cell in each time period. Several restrictions—a combination of linear and integer constraints—must be considered while performing this calculation:

1. Without intervention and without regard for fuel, the intensity of a cell one step into the future is the current intensity plus the sum of the intensities of the neighboring cells weighted by the transmission rate. If suppression team i is assigned to cell x at time $t-1$, the intensity is reduced. If the cell's fuel is below a certain value at time $t-1$, then the intensity is again reduced. Recall that the intensity of a cell is upper bounded by the initial fuel of that cell.
2. The remaining fuel at a particular point in time is a function of the intensity (intensity is assumed to be the fuel burned in a particular time period).
3. If there is insufficient fuel in cell x at period t, then the intensity of that cell in the next time point is 0. If there is sufficient fuel, then the intensity is at most $F_0(x)$, the initial fuel in the cell, which is already implied in the formulation.
4. Each suppression team or vehicle is assigned to at most one cell in each period.
5. The fuel and intensity are continuous nonnegative variables, and the sufficient fuel and team assignment variables are binary.

The mixed-integer linear optimization model has two sets of binary variables: the $A_t(x, i)$ variables that model the assignment of suppression teams i to cells x over time t, and the variables that model the loss of fuel over time.

In highly resource-constrained environments (when it is not possible to solve the above model to optimality), extremely good approximate solutions can still be obtained by relaxing the $A_t(x, i)$ variables to be continuous within the unit interval [0, 1]. Then, given an optimal solution with fractional values for the $A_t(x, i)$ variables at $t = 0$, we can compute a score value $v(x)$ for each cell x as $v(x) = \sum(A(x, i))$ over i. Suppression teams are then assigned to the cells with the highest values of the index v.

Calibrating the Optimization Model

Given the parameters for the original MDP formulation of the tactical wildland fire management problem, the parameters for our nominal optimization formulation are obtained as follows:

1. The intensity in a cell at time t is computed by a modified version of the fire dynamics that assumes no intervention and infinite fuel. Further, this approach assumes that transmission rates between cells are 1, which implies that the fire spreads as quickly as possible.
2. The initial fuel in a cell is obtained by summing the fuel threshold (minimum fuel capable of sustaining fire) and the intensity values over times $t = 0, 1, \ldots, \min\{T, F(x)\}$, where $F(x)$ is the number of periods that cell x can burn into the future, according to the original MDP dynamics.
3. Intuitively, since the intensity $I_t(x)$ can be thought of as how much fuel was consumed by the fire in cell x at time t, the initial fuel value $F_0(x)$ can be thought of as a limit on the cumulative intensity in a cell over the entire horizon. Once the cumulative intensity has consumed most of the fuel, the fuel in the cell enters the interval $[0, \delta]$, at which point the intensity is forced to zero for all remaining time periods.

Computational Experiments

This section presents experiments comparing MCTS and the MO formulation in order to understand their relative strengths and weaknesses. Two principal insights are obtained from the computations:

- Overall, MO performs as well as or better than MCTS performs, and for even reasonably large problems, the solution times are acceptable.
- Although the MCTS approach works well for certain smaller examples, its performance can degrade for larger examples (with a fixed computational budget).

Algorithmic Parameters and Experimental Setup

In what follows, we use a custom implementation of MCTS written in C++ and use the mixed-integer opti-

–8	–9	–10	–11	–12	–13	–14	–10
–7	–8	–9	–10	–11	–12	–13	–14
–6	–7	–8	–9	–10	–11	–12	–13
–5	–6	–7	–8	–9	–10	–11	–12
–4	–5	–6	–7	–8	–9	–10	–11
–3	–4	–5	–6	–7	–8	–9	–10
–2	–3	–4	–5	–6	–7	–8	–9
–1	–2	–3	–4	–5	–6	–7	–8

(a) k = 8

-12	-13	-14	-15	-16	-17	-18	-19	-20	-21	-22	-10
-11	-12	-13	-14	-15	-16	-17	-18	-19	-20	-21	-22
-10	-11	-12	-13	-14	-15	-16	-17	-18	-19	-20	-21
-9	-10	-11	-12	-13	-14	-15	-16	-17	-18	-19	-20
-8	-9	-10	-11	-12	-13	-14	-15	-16	-17	-18	-19
-7	-8	-9	-10	-11	-12	-13	-14	-15	-16	-17	-18
-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16	-17
-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16
-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15
-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13
-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12

(b) k = 12

FIGURE 5. In these graphs of the rewards for grid one, the magnitude of rewards increases from lower left to upper right (the upper right corner is fixed at –10 in each case). The general objective of fighting a fire in the grid one scenario is to prevent the fire from spreading to the upper right corner.

mization software Gurobi Optimizer 5.0 [19] to solve the MO formulation. All experiments were conducted on a computational grid with 2.2 GHz cores with 4 GB of RAM in a single-threaded environment. Although it is possible to parallelize many of the computations for each of the four methods, we do not explore this possibility in these experiments.

To ease comparison in what follows, we generally present the performance of each of our algorithms relative to the performance of a randomized suppression heuristic. At each time step, the randomized suppression heuristic chooses cells (without replacement) from those cells that are currently burning and assigns suppression teams to them. This heuristic should be seen as a naïve straw man for comparison only. We will also often include the performance of our more tailored heuristic, the FW heuristic, as a more sophisticated straw man.

Grid One

The primary purpose of this experiment is to explore the scalability of the various approaches.

In this setup, a k-by-k grid is generated with a varying reward function. There is a –1 reward received when the lower left cell is burning, and the reward for a cell burning increases by 1 when moving one box up or to the right across the grid. Also, the reward in the upper right-hand corner is always –10. Figure 5 shows the rewards for k = 8 and k = 12 grids. The fire in this experiment propagates with a probability of ignition from nearest neighbors of 0.06. For this experiment, suppression efforts are successful with an 80% probability—$Q(x)$ = 0.8 for all x in the grid space.

For a single simulation, we randomly generate an initial fire configuration (whether or not each cell is burning and what the fuel level is in each cell). After an initial fire is simulated, suppression then begins according to one of our four approaches with i teams. The simulation and suppression efforts continue until the fire is extinguished or the entire area is burned out.

A typical experiment will repeat this simulation many times with different randomly generated initial fire configurations and will aggregate the results. Table 1 shows summary statistics for the initial fire configurations used in this experiment.

The specific process for initializing the fire configuration in a simulation is as follows:

1. Initialize all of the cells with a fuel level and seed a fire in the lower left-hand cell.
2. Allow the fire to randomly propagate for a certain number of steps. Note that the lower left-hand cell will naturally extinguish at some point.

3. Next, scale the fuel levels by a factor of $k^{-0.25}$. This rescaling of the fuel levels is necessary to reduce the length of experiments when the number of suppression teams is insufficient to successfully fight the fire.

Grid Two

This experiment was designed to explore the ability of the various approaches to evaluate how the current allocation of suppression teams will impact the future reward received. Because the reward function may be very different outside the local region of the fire, the approaches may need to plan many steps into the future.

This setup mirrors the setup in the previous experiment with two exceptions. First, we initialize fires in the middle of the grid. Second, the reward function for cells is exponential across the grid. Specifically, at time $t = 0$ we ignite the cell in the middle of the grid. The reward for cell $x = (i, j)$ is proportional to $e^{-\lambda i}$ (the reward only depends on the horizontal location of the cell in the grid). In other words, cells located to the left are more valuable. The value of λ controls the rate at which the reward grows. Some typical reward curves are shown in Figure 6 for a $k = 20$ grid. Observe that for large values of λ, the local reward structure at the site of the fire may seem quite flat (e.g., for larger values of i). Good strategies need to account for the fact that, despite this local structure, suppression of cells on the right-hand side of the fire is ultimately more valuable than suppression to the left.

In this experiment, suppression efforts are still 80% successful. The spread probabilities $P(x, y)$ are 0.02 for y nearest neighbors of x; otherwise they are 0. As in the previous experiment, we begin with a random initial fire configuration. Table 2 shows summary statistics for the initial fire configurations randomly generated in this experiment.

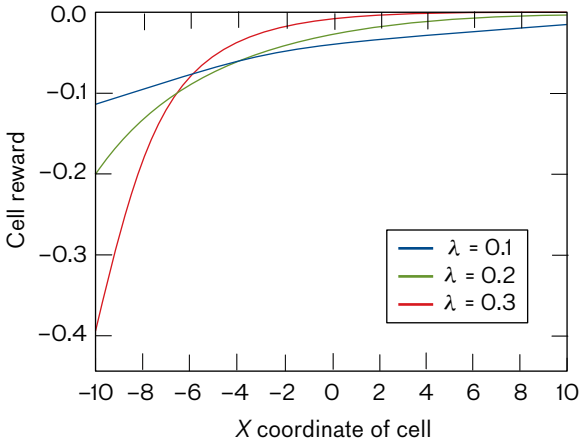

FIGURE 6. The reward structure for various values of λ is shown for grid two. When λ is large, it is more important to consider future fire trajectories because there is significantly more value in preventing the fire from spreading to the right-hand side than in minimizing the size of the fire in the left-hand side.

State Space Size

We first study the performance of our algorithms as the size of the state space grows. We simulate the performance of each of our methods on grid one with either four or eight suppression teams, using our default values of the hyperparameters and varying the grid size. For each algorithm and combination of parameters, we simulate 256 runs and amalgamate the results.

Figures 7a and 7b show the average and maximum solution time per iteration of the MO methodology when requesting at most 120 s of computation time. Notice that for most grids, the average time is well below the threshold; in these instances, the underlying integer program is solved to optimality. For some grids, though, a few iterations require much longer to find a feasible

Table 1. Grid One Initial Fire Statistics

	$k = 8$	$k = 12$	$k = 16$	$k = 20$	$k = 30$
Mean cells burning	37.6	91.4	168.7	275.5	664.2
Maximum cells burning	62	142	244	372	845
Mean fuel level for burning cells	15.8	19.9	22.8	25.7	31.4
Fuel level for nonburnt cells	24	29	34	38	46

Table 2. Grid Two Initial Fire Statistics

	k = 9	k = 17	k = 25
Mean cells burning	37.8	154.9	363.7
Maximum cells burning	69	224	487
Mean fuel level for burning cells	5.3	7.5	8.9
Fuel level for nonburnt cells	8	11	13

integer solution (cf. the long upper tail for k = 30 in Figure 7b). Consequently, we compare our MO to the MCTS methodologies with both 60 s and 120 s of computation time.

A summary of the results is seen in Figures 8a and 8b. We stress that the runs with four suppression teams are more difficult than those with eight teams; these instances are more resource constrained. Several features are evident in the plot. First, all three methods seem to outperform the FW heuristic, but there seems to only be a small difference between the two MCTS runs. The MO method does seem to outperform the MCTS method, especially for tail-hard instances—namely, when the lower whisker on the MO plot is smaller than the corresponding whisker on the MCTS plots.

To assess some of the statistical significance of these differences, we fit two additive-effects models to the data: one for eight suppression teams and one for four teams. In both cases, there are no significant second-order interactions. The results suggest that the two MCTS methods are very similar, with a slight advantage for the 120 s run, and that the MO method with a time limit of 60 s outperforms both.

To further understand the effect on performance of varying the time limit per iteration of the MCTS method, we reran the above experiment with 60 s, 90 s, and 120 s.

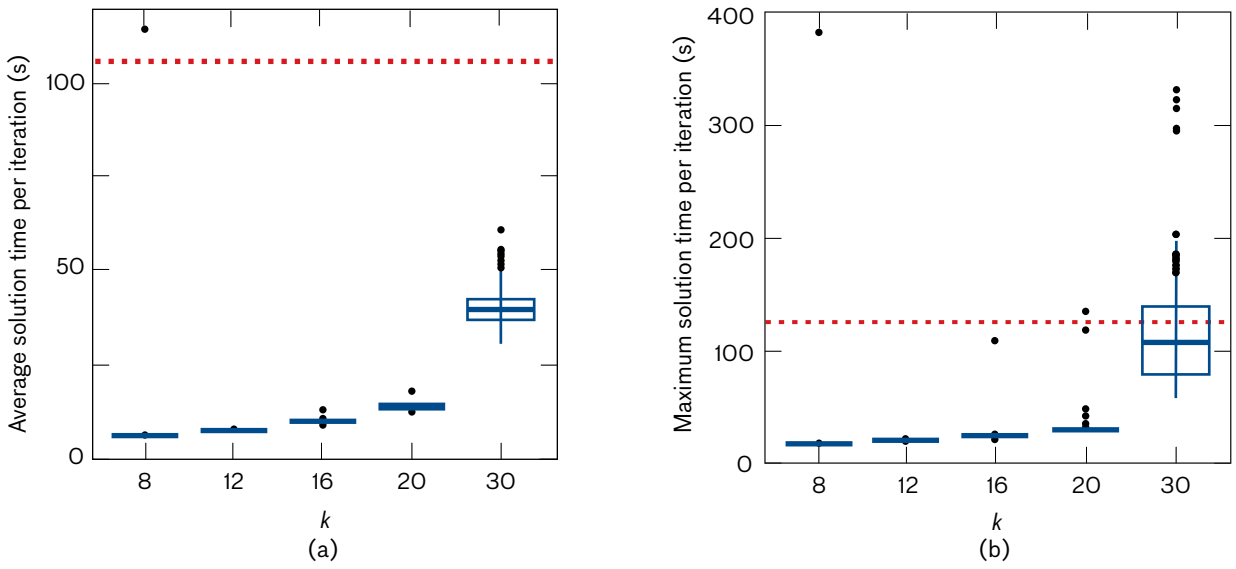

FIGURE 7. The average (a) and maximum (b) iteration solution times with eight teams are well below the desired time limit of 120 s (dotted line). Instances that exceed their allotted time are typically not solved to optimality. In this figure and in the subsequent similar figures, the boxplots summarize the data distribution using Tukey's conventions [20]. Each "box" spans from the first quantile to the third quantile, with the horizontal line indicating the median. The whiskers (top and bottom of boxes) are nonparametric estimates of the range of the data. Any outliers beyond the whiskers are plotted as points.

Results showed that the difference in performance is small. In summary, these results suggest that the MCTS algorithm performs comparably to the MO methodology as the state space grows large, but that the MO methodology does have a slight edge.

Action Branching Factor

Intuition suggests that the performance of the MCTS algorithm is highly dependent on the magnitude of the action branching factor, i.e., the number of actions available from any given state. Without progressive widening, when the action branching factor is larger than the number of iterations, the MCTS algorithm will only expand the search tree to depth one. Even with progressive widening, choosing good candidate actions is critical to growing the search tree in relevant directions. Consequently, in this section we study the performance of our algorithms with respect to the action branching factor.

We compute the relative improvement of the MCTS and MO approaches over the randomized suppression heuristic on grid one over 256 simulations with k = 10. Figure 8 summarizes the average relative improvement for each of our methods. Consider two scenarios: one with a fire being suppressed by a small team of resources and another by a large team of resources. In the case of the small team, intelligently allocating the resources is very important to extinguishing the fire because if the fire grows too large, the team will never be able to keep up with the spread of the fire. With more resources, their allocation is less critical. Thus, our methods are able to

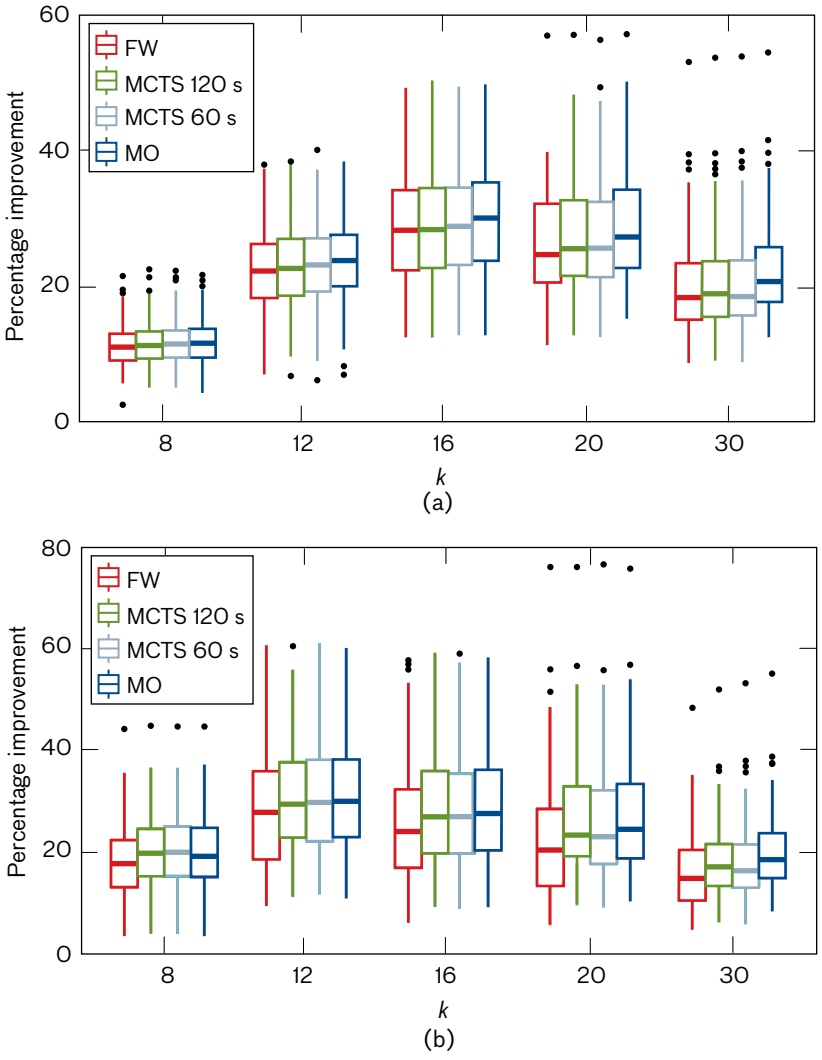

FIGURE 8. Performance is the percentage improvement over the random assignment of resources and is presented as a function of state space size. All three methods outperform the FW heuristic. The MO method seems to outperform the MCTS method, while there is little difference between the two MCTS runs. There are eight teams assigned in (a) and four teams in (b).

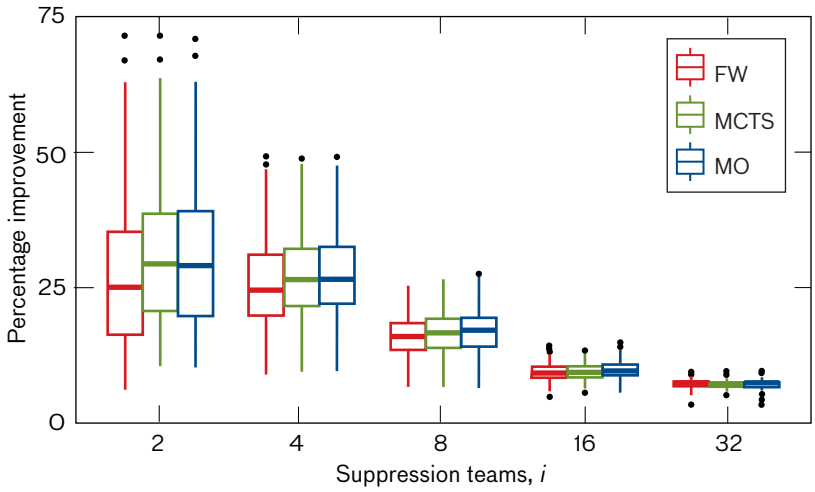

FIGURE 9. The number of suppression teams affects the performance. Note that the MCTS and MO methods perform better than FW when there are fewer suppression teams and that MO always performs better than MCTS. In general, fires are more difficult to fight with a simple heuristic algorithm when there are fewer suppression teams.

improve more upon random allocations with a small team than with a large team, a result that is most evident in the case shown in Figure 9.

Recall that it is not possible to control the exact time used by the MO algorithm. Figure 10 shows a boxplot of the average time per iteration for the MO approach. As shown in the plot, the results of the MCTS algorithm with 60 s of computational time are a fair comparison.

The relative performance of all computational methods degrades as the number of teams becomes large, principally because the randomized suppression heuristic improves with more teams. Although the FW heuristic is clearly inferior, the remaining approaches appear to perform similarly. Indeed, analysis of variance testing suggests the differences between MO and MCTS are not statistically significant. To try to isolate more significant differences between the methodologies, we reran the above experiment with $k = 20$. The results can be seen in Figure 11 and the average solution times in Figure 10b.

In contrast with the previous experiment, MO appears to outperform MCTS. Interestingly, although MCTS seems to outperform the FW heuristic for a small

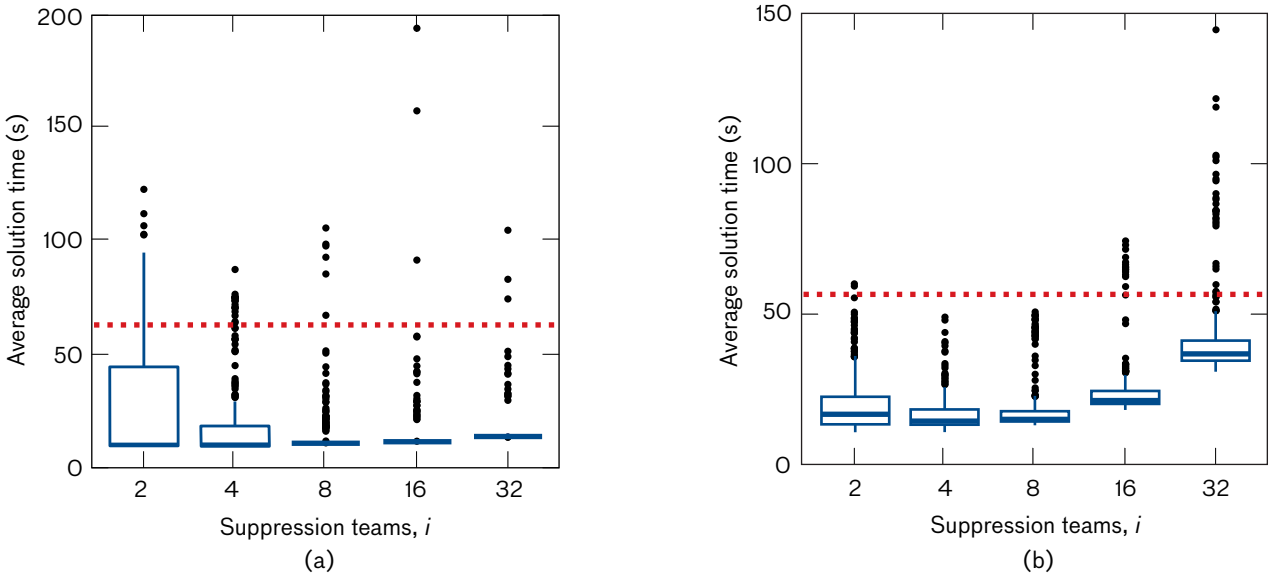

FIGURE 10. In many instances, MO can return a solution within a few seconds. However, in some cases, MO requires hundreds of seconds to return a feasible solution. Here, (a) has $k = 10$ and (b) has $k = 20$.

number of suppression teams, it performs worse than FW for more teams. To test the significance of these differences, we used a linear regression model for the improvement over the randomized suppression heuristic as a function of the number of teams, the algorithm used, and potential interactions between the number of teams and the algorithm used. However, MO outperforms FW for all team sizes with statistical significance while MCTS is statistically worse than FW with 16 or 32 teams.

In summary, differences between the MO and MCTS methods become visible only when the grid size k is large, i.e., when the instances are sufficiently difficult to solve. It appears that although progressive widening and the genetic algorithm for action selection partially address the challenges of a large action-state branching factor, the MO approach is better suited to these instances.

Asymmetric Costs and Horizon Length

Recall that in grid two the cost structure is asymmetric; cells to the right of the grid are more valuable than cells to the left. At the same time, the local reward structure at the point of ignition is relatively flat. A good algorithm must recognize the differential value despite the local reward structure.

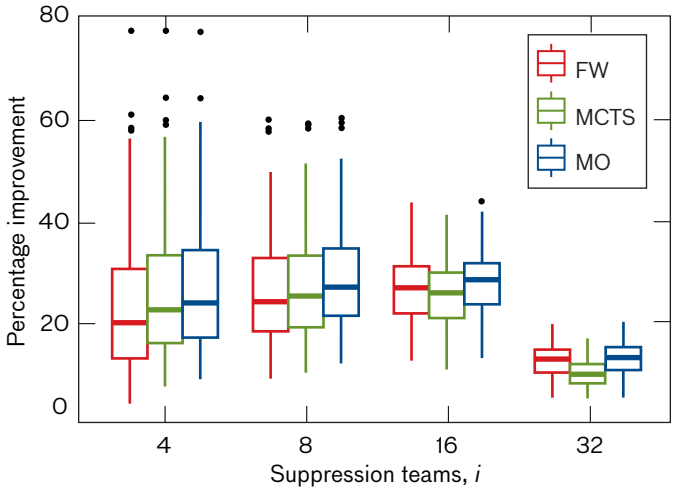

FIGURE 11. Performance is affected by the number of suppression teams (k = 20). MCTS performs better than the FW heuristic for fewer teams, while MO outperforms FW for all team sizes tested.

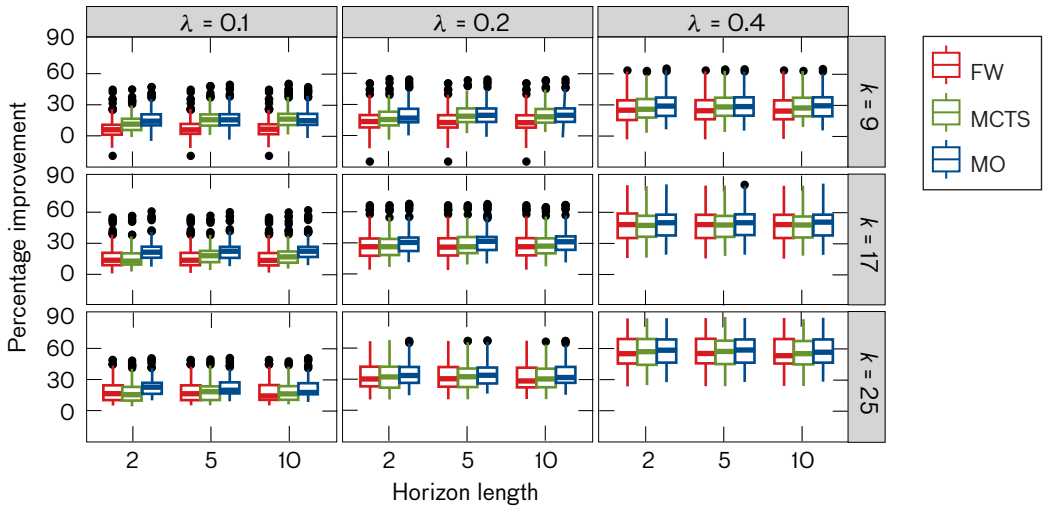

FIGURE 12. Different panels correspond to different values of $k \in \{9, 17, 25\}$ and $\lambda \in \{0.1, 0.2, 0.4\}$. MO performs best compared to the other methods for small λ values, while the difference is less for larger values. The horizon length is the number of boxes (beyond the nearest neighbors) that are included in the analysis.

We consider different combinations of exponential scaling factor λ, grid size k, and horizon length (the amount of time into the future that the algorithms consider for each decision step). Figure 12 summarizes the performance of each of our methods. For all the methods, there is an upward trend as λ increases. For small values of λ (i.e., flatter reward structure), MO has a marked edge over the other methods. As λ increases, the difference shrinks.

Looking Forward

A number of future directions emerge. A significant research effort remains to incorporate parameters informed by real wildfire data into the model. Once the model is properly calibrated, it can be used to compare the effectiveness of the methods described here against the performance of decision makers experienced in wildfire suppression. The appendix describes a prototype decision support tool that could be used to inform decision makers.

Several important issues also merit further investigation. Although we attempted to understand how the hyperparameters of MCTS affect its performance, our results show that this relationship is complex, and further investigation is needed to better elucidate it. Regarding the optimization-based methods, our MO formulation is only one way to approximate the discrete, stochastic dynamics of the true Markov decision process. Lastly, it may be interesting to consider a hybrid method in which the MO formulation and the MCTS approach are combined. For example, the MO formulation may be used to guide action generation or as part of the rollout heuristic within MCTS. Such a hybrid approach could improve on both of the pure approaches.

Acknowledgments

The authors would like to thank Michael Boulet, Jason Thornton, and the rest of Lincoln Laboratory's Autonomous Systems Line committee for supporting and guiding the research reported on in this article. ■

References

1. R. Gorte, "The Rising Cost of Wildfire Protection," Headwater Economics, Technical Report, 2013, available at http://headwaterseconomics.org/wphw/wp-content/uploads/ fire-costs-background-report.pdf.
2. J. McLennan, A.M. Holgate, M.M. Omodei, and A.J. Wearing, "Decision Making Effectiveness in Wildfire Incident Management Teams," *Journal of Contingencies and Crisis Management,* vol. 14, no. 1, 2006, pp. 27–37.
3. G. Papadopoulos and F.N. Pavlidou, "A Comparative Review of Wildfire Simulators," *IEEE Systems Journal,* vol. 5, no. 2, 2011, pp. 233–243.
4. C. Tymstra, R.W. Bryce, B.M. Wotton, S.W. Taylor, and O.B. Armitage, "Development and Structure of Prometheus: The Canadian Wildland Fire Growth Simulation Model," Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre Information, Report NOR-X-417, 2010.
5. M.A. Finney, "FARSITE: Fire Area Simulator–Model Development and Evaluation," U.S. Department of Agriculture (USDA) Forest Service, Research Paper RMRS-RP-4, 2004.
6. D. Boychuck, W.J. Braun, et al., "A Stochastic Forest Fire Growth Model," *Environmental and Ecological Statistics,* vol. 1, no. 1, 2008, pp. 1–19.
7. L. Ntaimo, J.A.G. Arrubla, C. Stripling, J. Young, and T. Spencer, "A Stochastic Programming Standard Response Model for Wildfire Initial Attack Planning," *Canadian Journal of Forest Research,* vol. 42, no. 6, 2012, pp. 987–1001.
8. J.S. Fried, J.K. Gilless, and J. Spero, "Analyzing Initial Attack on Wildland Fires Using Stochastic Simulation," *International Journal of Wildland Fire,* vol. 15, 2006, pp. 135–146.
9. R.C. Rothermel, "A Mathematical Model for Predicting Fire Spread in Wildland Fuels," USDA Forest Service, Technical Report INT-115, 1972.
10. U.S. Department of Agriculture, Wildland Fire Suppression Tactics Reference Guide, 1996.
11. S. Martin-Fernández, E. Martínez-Falero, and J.M. Pérez-González, "Optimization of the Resources Management in Fighting Wildfires," *Environmental Management,* vol. 30, 2002, pp. 352–364.
12. K.G. Hirsch, "Canadian Forest Fire Behavior Prediction (FBP) System: User's Guide," Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre, Special Report 7, 1996.
13. C.B. Browne, E. Powley, et al., "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 4, no. 1, 2012, pp. 1–43.
14. A. Couëtoux, J.B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous Upper Confidence Trees," *Proceedings of the 5th International Conference on Learning and Intelligent Optimization,* 2011, pp. 433–445.
15. D. Whitley, "A Genetic Algorithm Tutorial," *Statistics and Computing,* vol. 4, no. 2, 1994, pp. 65–85.
16. R.W. Floyd, "Algorithm 97: Shortest Path," *Communications of the ACM,* vol. 5, no. 6, 1962, p. 345.
17. F. Avram, D. Bertsimas, and M. Ricard, "Fluid Models of Sequencing Problems in Open Queueing Networks: An Optimal Control Approach," in *Stochastic Networks: Proceedings of the International Mathematics Association,* F.P. Kelly and R.J. Williams, eds., vol. 71, pp. 199–234. New York: Springer, 1995.
18. G. Gallego and G. Van Ryzin, "Optimal Dynamic Pricing of Inventories with Stochastic Demand over Finite Horizons," *Management Science,* vol. 40, no. 8, 1994, pp. 999–1020.
19. Gurobi Optimization Inc., Gurobi Optimizer Reference Manual, 2013, available at http://www.gurobi.com.
20. R. McGill, J.W. Tukey, and W.A. Larsen, "Variations of Box Plots," *The American Statistician,* vol. 32, no. 1, 1978, pp. 12–16.

Appendix

Wildland Fire Decision Support Tools

To evaluate and demonstrate the performance of the algorithms from the perspective of the end user, we built a decision support tool for wildland fire incident commanders. The tool was built using an implementation of the National Aeronautics and Space Administration's (NASA) open-source geographical World Wind application programming interface (API), shown in Figure A1. This API allowed us to simulate stochastic wildland fires across a three-dimensional globe. The tool can import a variety of different data layers that inform the wildfire model and can be used as a training program for the pre-evaluation of scenarios in highly dangerous areas.

Concept of Operations

An incident commander would use this software during a wildland fire to simulate the probabilistic direction of a spreading fire and the location of future flare-ups. The commander either shift-clicks anywhere on a map and ignites a fire or free draws any shaped fire and instantly gathers information, such as the perimeter length and fire area (shown in Figure A2). The incident commander's knowledge of the local region may include information about past fires that may lead to improved selection of initial flare-up locations rather than to the selection of random locations.

The wildland fire simulator uses an adaptable grid structure to display the spread. Once a fire is placed and zero resources are set, the user plays the simulation, which is driven by the transition probability equation, and watches the fire spread. Each cell is colored by a scale ranging from yellow (full fuel) to red (near-zero fuel). The user can pause the simulation at any time and go back and forth between steps in the fire progression and in the suppression efforts or reinitiate fire locations and fuel levels. During the active simulation, the user is able to easily change the dynamics of the fire. Under the simulation settings option, the user sets the time interval, the size of the cells, and the wind direction.

Once a wildland fire situation has been created, a user inputs the quantity and the type of resources at his or her disposal. Included are three air resources (airplanes, helicopters, and unmanned aerial vehicles) and three ground resources (fire engines, hotshot crews [readiness teams], and bulldozers) that can be allocated for fire suppression. We focused on applying only air resources to the fire, giving us a wider range of actions because those actions are not restricted to the fire perimeter (as is the case with ground crews working near the fire boundaries). After the resources are set, the user plays the simulation and watches as the resources are allocated by the recommendations of the algorithms. At any time, a user can pause the resource allocation and change algorithm parameters.

Data Acquisition

The user places water, population, and different vegetation types onto the map. A major factor in how incident commanders allocate their resources is population. Populated areas (shown on the left in Figure A2 and in three corners in Figure A3) have a much higher cost (reduced rewards) associated with them.

Serialization

This software allows the user to see the output of the implemented algorithms and to experiment with the results. The scenarios can be saved to a JavaScript Object Notation file and loaded back into the simulator at a later time. Every step of the simulation is saved so a user can review the scenario from start to finish. Saving each wildland fire map is useful for continuous tests on a single situation. Each simulation can be saved as a video file of the fire spread, resource placement, and fire suppression.

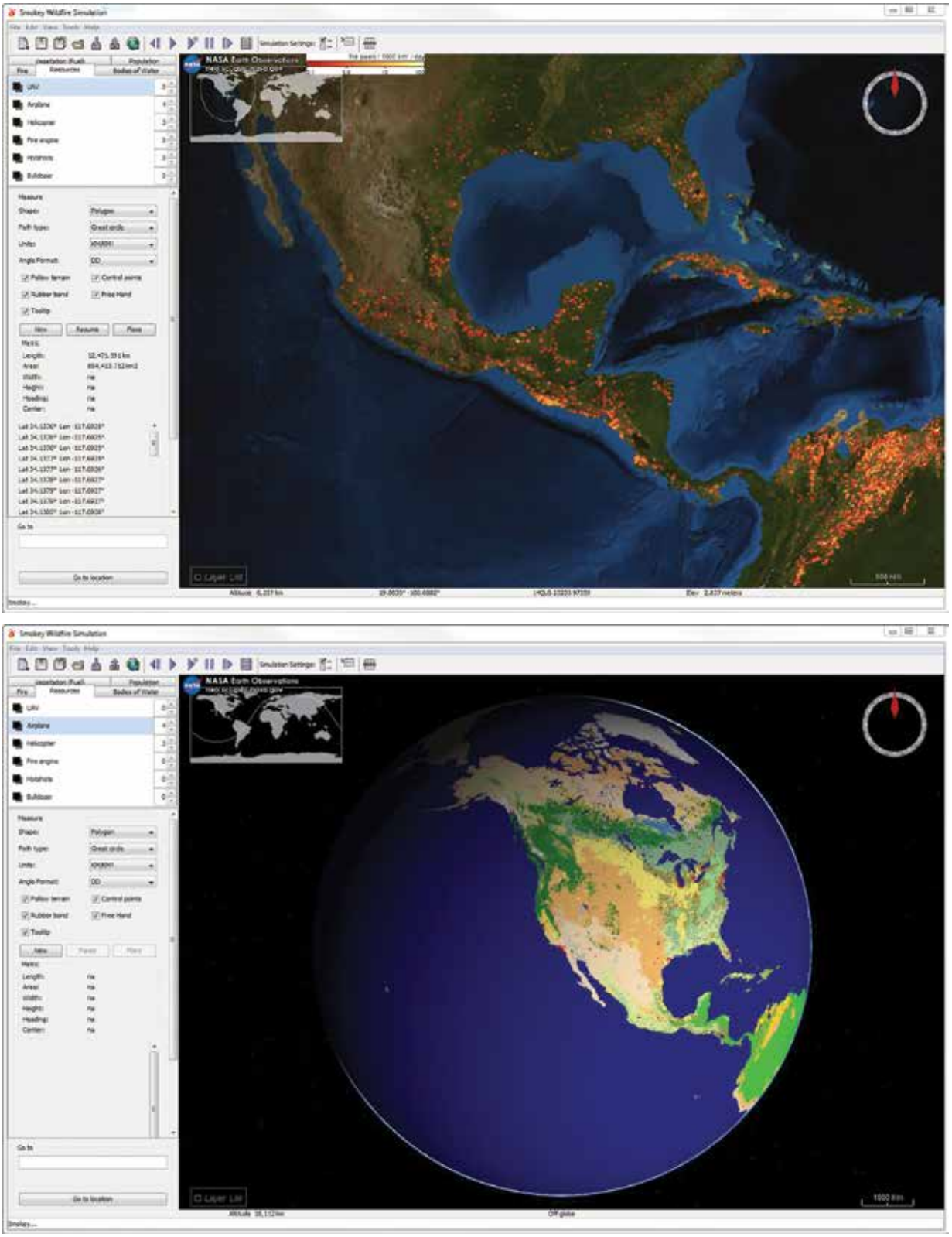

FIGURE A1. Active wildland fires (red/orange areas in upper image) and land-cover classification layers (varying patches of color in lower image) are shown in these screenshots. NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) satellite often updates layers, such as land cover maps, wildland fire locations, population density, and vegetation index.

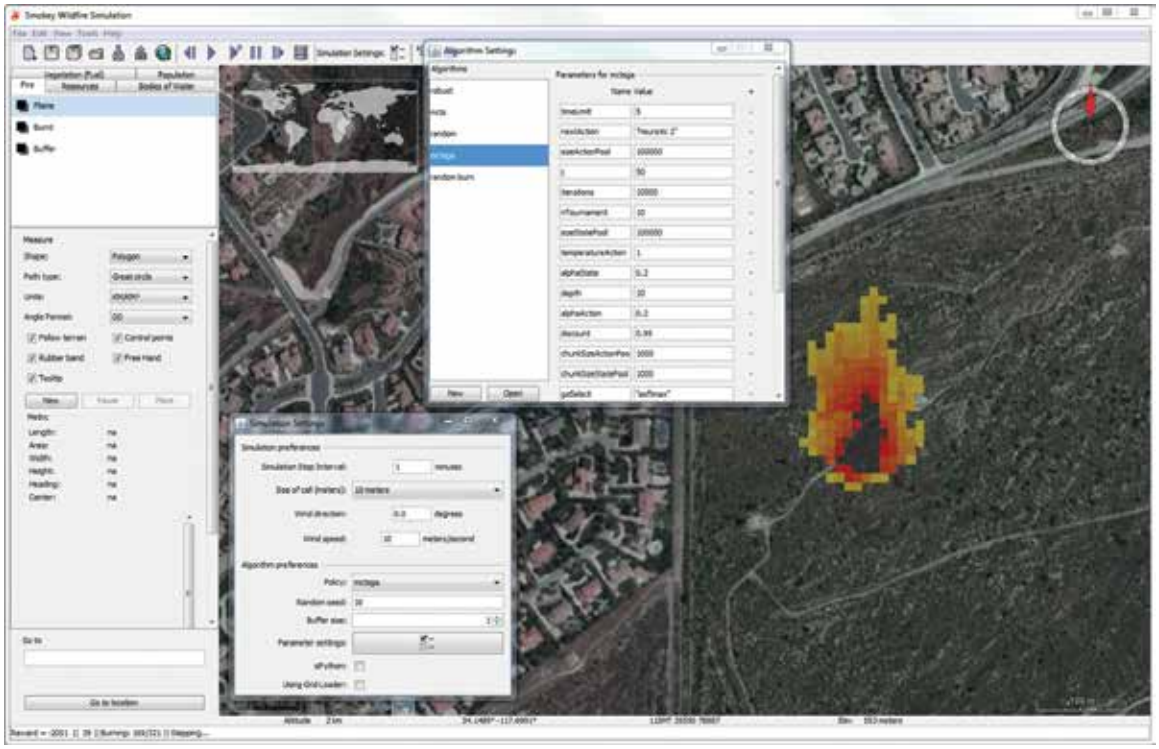

FIGURE A2. Simulation settings and algorithm parameter settings allow a user to create different specific scenarios of wildland fires.

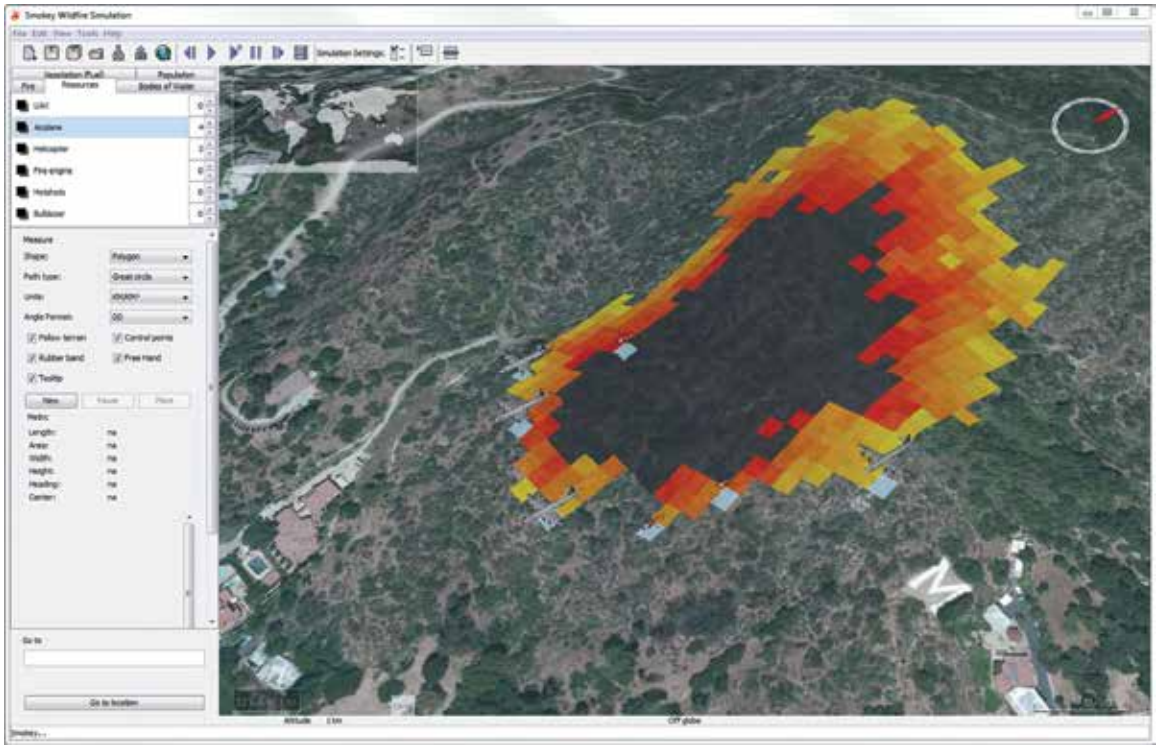

FIGURE A3. Air resources are allocated to a simulated wildland fire. Blue grids (in several points in the lower left of the burning region) are those locations where suppressants are applied (airplane symbols above).

About the Authors

J. Daniel Griffith is an associate staff member in the Intelligence and Decision Technologies Group. His research focuses on applying information and decision theory to improve the exploitation of sensor data to support military and intelligence decision making. He originally joined Lincoln Laboratory in 2007 as a member of the Surveillance Systems Group, where his work involved simulation and analysis of sense-and-avoid systems for unmanned aircraft. He received bachelor's and master's degrees in aeronautics and astronautics from MIT.

Mykel J. Kochenderfer is an assistant professor of aeronautics and astronautics at Stanford University and director of the Stanford Intelligent Systems Laboratory. Prior to joining the faculty at Stanford in 2013, he was a member of the technical staff at Lincoln Laboratory, working on airspace modeling and aircraft collision avoidance. He received a 2011 MIT Lincoln Laboratory Early Career Technical Achievement Award for these efforts. He holds bachelor's and master's degrees in computer science from Stanford University and a doctorate from the University of Edinburgh. He is a third-generation pilot.

Robert J. Moss is an assistant staff member in the Surveillance Systems Group. He joined Lincoln Laboratory as a co-op student while earning his bachelor's degree in computer science from the Wentworth Institute of Technology. At Wentworth, his research included modeling the rotation curve of galaxies. His current research focuses on the development and analysis of the next-generation airborne collision avoidance system.

Velibor V. Mišić is an assistant professor at the Anderson School of Management at the University of California Los Angeles. He earned his doctorate from the Operations Research Center in the MIT Sloan School of Management. He received his bachelor's degree in industrial engineering in 2010 and a master's degree in industrial engineering in 2012, both from the University of Toronto. His research is focused on developing new methods for large-scale dynamic decision making under uncertainty and investigating problems at the intersection of marketing and operations management.

Vishal Gupta is an assistant professor of data science and operations at the Marshall School of Business of the University of Southern California. He received his bachelor's degree in mathematics and philosophy from Yale University, master's degree in mathematics from the University of Cambridge, and doctoral degree in operations research from MIT. His doctoral studies focused on using optimization techniques to create new, tractable models for uncertainty and behavior.

Dimitris Bertsimas is the Boeing Leaders for Global Operations Professor of Management, a professor of operations research, and the codirector of the Operations Research Center at the Sloan School of Management at MIT. He has been a member of MIT since 1988. His research interests include optimization, stochastic systems, machine learning, and their applications. In recent years, he has worked in robust optimization, statistics, health care, and finance. He was a cofounder of Dynamic Ideas LLC, which developed portfolio management tools for asset management. He is also the founder of Dynamic Ideas Press, a publisher of scientific books. He holds a bachelor's degree in electrical engineering and computer science from the National Technical University of Athens, Greece, and a master's degree in operations research and a doctorate in applied mathematics and operations research from MIT.