A Study of Evacuation Planning for Wildfires

Christian Artigues[1], Emmanuel Hebrard[1], Yannick Pencole[1], Andreas Schutt[2,3], and Peter J. Stuckey[2,3]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2] Decision Sciences, Data61 CSIRO, Melbourne, Australia
[3] Department of Computing and Information Systems, The University of Melbourne, Melbourne, Australia

Abstract. The GEO-SAFE project gathers researchers and fire emergency practitioners from EU and Australia with the aim to design innovative models and efficient response tools based on optimization methods for fighting wildfires. In this paper, we consider an evacuation planning problem issued from discussions with practitioners, where a wildfire is threatening a region with intermediate populated centers. As in earlier approaches in case of a flood, we use a constraint optimization model involving *malleable* tasks to represent the evacuation of a population and a `cumulative` constraint per route segments. Indeed, in order to mitigate congestion risks, the authorities may delay the start of the evacuation but they may also affect the rate of evacuation by modulating the method used to raise the alarm. However, we consider a different objective: we maximize the minimum "safety margin", weighted by the population, over every road segment. We introduce a new heuristic and a global flow constraint propagator. Moreover, we also propose an instance generator based on a random generation of road networks and basic fire propagation models. This generator produces challenging benchmarks even with very few evacuation tasks. Finally, we report the results of extensive computational experiments done using CP Optimizer.

1 Introduction

In EU and Australia, every year thousands of square miles of forests and other lands burn due to wildfires. These fires cause important economic and ecological losses, and often, human casualties as for instance during the Black Saturday bushfires across the Australian state of Victoria in February 2009 [13]. The overall objective of the GEO-SAFE project [7] is to develop methods and tools enabling to set up an integrated decision support system to assist authorities in optimizing the resources during the response phase to a wildfire (fire suppression, life and goods protection). One critical and crucial part of this integrated decision support is the ability to perform large-scale evacuation planning. As detailed in [14], there are commonly three categories of evacuees: the ones that leave early, the ones that shelter in refuge and the ones who stay at their properties and fight. This work focuses on the evacuation of this third group (late evacuees) which is called the *late evacuation planning problem*.

While in practice most evacuation planning is principally designed by experts using simple heuristic approaches or scenario simulations [16], more recently optimization approaches to evacuation planning have been addressed, using a variety of optimization technology [2]. The usual test case for most of this work is flood evacuation planning [11, 5, 8]. Given accurate measurements of rainfall and topology, flood evacuation planning can make use of very accurate predictions of future water levels, and therefore, has a very accurate model of what infrastructure will be available at each stage of the evacuation.

Evacuation planning in case of wildfires is much harder. Wildfire propagations are inherently less predictable than floods. While flood levels mostly rely on the fixed topology of the area and rainfalls, wildfire mainly depends on the wildland fuels [12, 1], on the slope of the burning ground and more importantly on the speed and direction of the wind that can suddenly change at any time [17, 15]. Therefore, evacuation planning dedicated to wildfires must be much more robust to different future scenarios. A good evacuation plan in case of wildfire must not only minimize the evacuation time of the population but also maximize the spatial and temporal safety margin between the evacuees and the actual or potential wildfire front.

In this paper, we consider that the authorities already identified the accessible routes and shelters and estimated the population of the late evacuees. Evacuation takes place in individual vehicles and each center has a known population and a single predefined suggested evacuation route. All routes issued from each center converge toward a safe place, so congestion may appear on route segment shared by several evacuation paths. The interest of convergent evacuation plans has been underlined by several studies such as in [4] since it avoids congestion issued by driver slow-downs at forks. Furthermore, fire propagation models give a deadline on each route segment beyond which taking this segment comes at a high risk. To mitigate these risks, the authorities may delay the evacuation start time and rate for each center. Indeed, in practice, besides the possibility of assigning a start time to each evacuated zone, the authorities can also mobilize different levels of resources to increase the evacuation rate (e.g. number of agents knocking on people's door), to which people answer according to a behavioral model abstracted by a response curve [10]. Based on these concepts, the evacuation planning model is close to the one proposed in [5] called the non-preemptive evacuation planning problem (NEPP) in the context of a flood. In contrast with previously proposed models, the NEPP considers non-preemptive evacuation. Once the evacuation of a zone has started, it cannot be interrupted. Indeed, considering preemptive evacuation would make the problem much easier to solve but is hard to implement in practice. In case of wildfire, this would notably cause undesirable stress on evacuees. The major difference of the model considered in this paper with the NEPP model lies in the objective function. In [5], the main objective was to maximize the number of evacuees, and a secondary objective was to minimize the makespan, i.e. the total evacuation time, while enforcing deadlines on route segments. In case of a wildfire, the high variability in the fire propagation makes it necessary to avoid taking a route segment close

to the expected deadline. Hence, we consider a single objective by maximizing the minimum gap for all evacuated zones and all route segments between the deadline and the time by which the last evacuee leaves a route segment, weighted by the population of the evacuated zone.

2 The evacuation planning problem

We are given a tree $\mathcal{G} = (\mathcal{E} \cup \mathcal{T} \cup \{r\}, \mathcal{A})$ rooted in r standing for the evacuation routes from *evacuation nodes* $\mathcal{E}$ (leaves) to the *safe node* r through *transit nodes* $\mathcal{T}$. Every leaf/evacuation node $v \in \mathcal{E}$ is associated with a population count w_v. Every arc u, u' has a length $l_{uu'}$ and a capacity non-ambiguously denoted q_u as $\mathcal{G}$ is a tree. Moreover, the length of the path from a node u to a node u' is also written $l_{uu'}$.

Let $\mathcal{H} = [0, H]$ be the time span of the evacuation. We want to associate every leaf/evacuation node $v \in \mathcal{E}$ to a real s_v representing the delay of the evacuation notice and to a "response curve" ϕ_v describing the evacuation flow out of node v over time (starting from s_v). Population flows out of leaf/evacuation node v at rate $\phi_v(t)$ that is zero before time $s_v \geq 0$, and such that:

$$\int_0^H \phi_v(t)dt = w_v$$

Let $p(u)$ denote the parent of u in $\mathcal{G}$, $\hat{p}(u)$ its ascendants, $C(u)$ its children, $\hat{C}(u)$ its descendants and $L(u)$ those of its descendants that are leaves of $\mathcal{G}$. We assume that the evacuees never stop, so the flow in the downstream arcs can be computed by summing the flows in the incoming arcs and the flow $\phi_u(t)$ for any arc $\in \mathcal{E} \cup \mathcal{T}$ at any time $t \in \mathcal{H}$ is

$$\phi_u(t) = \sum_{v \in L(u)} \phi_v(t - l_{uv} - s_v)$$

In this paper, we consider as in [5] a simple response curve. The flow out of an evacuation node $v \in \mathcal{E}$ is a continuous decision variable and remain constant during the evacuation process, that is, the flow out of node $v \in \mathcal{E}$ is equal to h_v within the time interval $[s_v, e_v]$, with $e_v = s_v + \frac{w_v}{h_v}$ and zero otherwise. Therefore, the flow out of a node $u \in \mathcal{E} \cup \mathcal{T}$ at time $t \in \mathcal{H}$ is:

$$\phi_u(t) = \sum_{v \in L(u), s_v + l_{uv} \leq t < s_v + l_{uv} + \frac{w_v}{h_v}} h_v$$

It follows that the considered evacuation planning problem can be formally defined as the following constraint optimization problem.

Variables: We have a set of $|\mathcal{E}|$ non-preemptive tasks, one for every evacuation node. For every task standing for an evacuation node $v \in \mathcal{E}$ we need two variables, one for the constant rate $h_v \in]0, q_v]$ at which the evacuation will proceed and one for its start time $s_v \in [0, H - \frac{w_v}{q_v}]$ at which the evacuation will start.

Constraints: We have a single type of constraints to avoid "jams", i.e., flow exceeding the capacity of an arc. For each node $u \in \mathcal{T}$, we have a `cumulative` resource constraint of capacity q_u ensuring that $\phi_u(t) \leq q_u$, which is written:

$$\sum_{v \in L(u), s_v + l_{uv} \leq t < s_v + l_{uv} + \frac{w_v}{h_v}} h_v \leq q_u, \quad \forall u \in \mathcal{T}, \forall t \in \mathcal{H}$$

Objective: A due date d_u standing for the time at which the next road portion becomes unsafe is associated to every transit node u. The objective is to minimize the maximum lateness of any task, i.e., the difference between the time at which it leaves a node u and d_u, weighted by the population. Hence the objective is:

$$\min \max_{u \in \mathcal{T}, v \in L(u)} s_v + h_v/w_v + l_{uv} - d_u$$

Dominance rules and problem formulation A first observation is that we can simplify the objective function by retaining the transit node u minimizing $d_u - l_{uv}$. Let $d_v = \min_{u \in \mathcal{T}}\{d_u - l_{uv}\}$ denote this value.

The objective can therefore be rewritten as the maximum of $|\mathcal{E}|$ expressions:

$$\min \max_{v \in \mathcal{E}} s_v + h_v/w_v - d_v$$

The second following observation, also made in [5], allows to reduce the number of `cumulative` constraints to consider.

Observations 1 *For any two transit nodes $u, u' \in \mathcal{T}$, if $q_u \leq q_{u'}$ and $u' \in \hat{C}(u)$, then a jam in u' entails a jam in u.*

We thus only need to check jams in nodes u such that $\forall v \in \hat{p}(u),\ q_v > q_u$. In practice, it means that given an evacuation node $v \in \mathcal{E}$ we can explore the nodes in the route from v to r in reverse order, and for every stretch of road without branch, keep only the arc of minimal capacity, called the critical arc. Let $\tilde{\mathcal{T}}$ denote the reduced set of critical transit nodes to consider. The problem can thus be formulated as follows.

$$\text{minimize} \quad \max_{v \in \mathcal{E}} w_v\{s_v + h_v/w_v - d_v\} \tag{1}$$

$$\text{subject to:} \quad \sum_{v \in L(u), s_v + l_{uv} \leq t < s_v + l_{uv} + \frac{w_v}{h_v}} h_v \leq q_u, \quad \forall u \in \tilde{\mathcal{T}}, \forall t \in \mathcal{H} \tag{2}$$

$$h_v \in]0, q_v], s_v \in [0, H - \frac{w_v}{q_v}], \quad \forall v \in \mathcal{E} \tag{3}$$

3 Baseline approach using `cumulative` constraints

In [5], the NEPP was modeled using standard `cumulative` constraints. We consider this model for our problem as the baseline approach. Let $\overline{x}$ (resp. $\underline{x}$) denote

the largest (resp. smallest) value in the domain of a variable x. Given a set of tasks J with start time variable $s_i \in [\underline{s}_i, \overline{s}_i]$, processing time variable $p_i \in [\underline{p}_i, \overline{p}_i]$, height variable $h_i \in [\underline{h}_i, \overline{h}_i]$ and a resource r of constant capacity q_r, recall that $\texttt{cumulative}((s_i, p_i, h_i)_{i \in J}, q_r)$ enforces the relations

$$\sum_{i \in J | s_i \leq t < s_i + p_i} h_i \leq q_r \quad \forall t \in \mathcal{H}$$

Consequently, to model the problem, it suffices to associate a task v to each evacuation node $v \in \mathcal{E}$, with height variable equal to the rate $h_v \in]0, q_v]$, start time variable $s_v \in [0, H - \frac{w_v}{q_v}]$, completion time variable $e_v \in [\frac{w_v}{q_v}, H]$, and to duplicate and translate this task for each critical transit node u on its path towards the safe node. For each critical arc $u \in \tilde{\mathcal{T}}$, let i_{uv} denote the duplicate for evacuation node $v \in L(u)$. A resource is defined per critical arc $u \in \tilde{\mathcal{T}}$, with capacity q_u.

The baseline constraint program for the evacuation planning problem is obtained by replacing constraints (2) in the problem formulation by:

$$\texttt{cumulative}((s_{i_{uv}}, e_{i_{uv}} - s_{i_{uv}}, h_{i_{uv}})_{v \in L(u)}, q_u) \qquad \forall u \in \tilde{\mathcal{T}} \quad (4)$$

$$w_v = h_v(e_v - s_v) \qquad \forall v \in \mathcal{E} \quad (5)$$

$$s_{i_{uv}} = s_v + l_{uv} \qquad \forall u \in \tilde{\mathcal{T}}, \forall v \in L(u) \quad (6)$$

$$e_{i_{uv}} = e_v + l_{uv} \qquad \forall u \in \tilde{\mathcal{T}}, \forall v \in L(u) \quad (7)$$

$$h_{i_{uv}} = h_v \qquad \forall u \in \tilde{\mathcal{T}}, \forall v \in L(u) \quad (8)$$

In the model above, the information that tasks have a fixed energy is lost to the `cumulative` constraint propagator. Given a task v with energy w_v start time s_v, completion time e_v and rate h_v, the algorithm will consider a task of duration $\frac{w_v}{\overline{h}_v}$ and height $\frac{w_v}{\overline{e}_v - \underline{s}_v}$. When the upper bounds on consumption and duration are large, these values tend to 0, thus greatly hindering constraint propagation.

Example 1. Consider a resource of capacity 4 and four tasks as shown in Table 1. Given the total energy (second column) possible ranges for the rates, minimum start times and maximum completion times (3rd to 5th columns), bounds consistency on the duration and demand variables yields the ranges shown in the 6th and 7th columns, respectively.

Therefore, a standard `cumulative` constraint will use the lower bounds and consider four tasks of durations and consumptions 2, $\frac{3}{2}$, 1 and 1, respectively. The classic `cumulative` constraint will not adjust the domains further, as shown in Figure 1. The set of 3 solutions illustrated in this figure shows that every bound of every one of the eight variables (start times and height for each of the four evacuation tasks) is consistent.[1]

[1] Tasks v_4 and v_3 are symmetric so we omit the bounds for v_4

Table 1: Task parameters

	w_i	q_i	$\underline{s}_i$	$\overline{e}_i$	duration	height
v_1	6	3	0	3	$[2, 3]$	$[2, 3]$
v_2	6	4	1	5	$[\frac{3}{2}, 4]$	$[\frac{3}{2}, 4]$
v_3	4	4	1	5	$[1, 4]$	$[1, 4]$
v_4	4	4	1	5	$[1, 4]$	$[1, 4]$

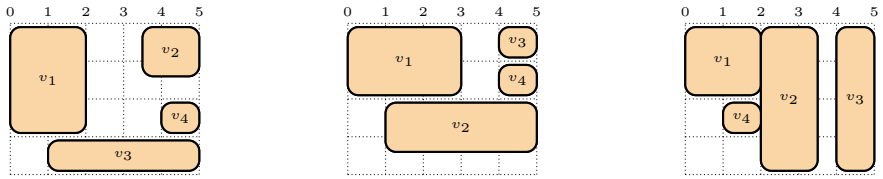


Fig. 1: Some feasible schedules

4 The `energetic_cumulative` constraint

The standard `cumulative` constraint is weak for this class of problems since it is unable to reason about total energy of each task. We can take advantage of the knowledge that the product (duration × rate) is a constant by using a more global constraint.

Given a set of tasks J with start time $s_i \in [\underline{s}_i, \overline{s}_i]$, completion time $e_i \in [\underline{e}_i, \overline{e}_i]$, height $h_i \in [\underline{h}_i, \overline{h}_i]$, constant energy w_i and a resource r of constant capacity q_r, `energetic_cumulative`$((s_i, e_i, h_i, w_v)_{i \in J}, q_r)$ enforces the relations:

$$\sum_{i \in J | s_i \leq t \leq e_i} h_i \leq q_r \quad \forall t \in \mathcal{H}$$
$$w_i = h_i(e_i - s_i) \quad \forall i \in J$$

Finding a support for this constraint is NP-hard since the particular case where the variables h_i are fixed is a `cumulative` constraint. However, because of the malleable nature of the tasks, the problem becomes easier under some assumptions. In particular, we consider here that we do not have minimum values on the height variable h_i of any task i, and we denote this particular case `energetic_cumulative`$\backslash\{\underline{h}\}$. We consider three further relaxations:

Let "r" denote the release dates, whose relaxation means that $\underline{s_i} = 0$ for every task $i \in J$; "d" denote the due dates whose relaxation implies $\overline{e_i} = H$ for every task $i \in J$ and some constant H; "$\overline{h}$" denote the maximum height whose relaxation implies $\overline{h_i} = q_r$ for every task $i \in J$. We write `energetic_cumulative`$\backslash\{S\}$ for the case where the subset of constraints $S \subseteq \{\underline{h}, \overline{h}, r, d\}$ are relaxed.

Theorem 1. *energetic_cumulative*$\backslash\{\underline{h}, x, y\}$ *is in P for any* $x \neq y \in \{r, d, \overline{h}\}$

Proof (sketch). Because of the lack of space, we give only the algorithm for `energetic_cumulative`$\backslash\{\underline{h}, r, d\}$ as the other two are trivial: without bound on the height, we can vertically slice the tasks as thin as possible while satisfying the capacity q_r and stack them w.r.t. their release or due dates (depending on which was relaxed). Algorithm 1 schedules the tasks so that every task starts at time 0 and the latest completion time is minimum while satisfying $h_i \leq \overline{h_i}\ \forall i \in J$ Thus, the constraint is satisfiable iff this latest completion time is less than H.

Algorithm 1: `UsageSorted`

W.L.O.G, let $J = \{1, \ldots, n\}$ be such that $i < j \implies \frac{w_i}{q_i} \geq \frac{w_j}{q_j}$;
$K \leftarrow \sum_{i=1}^{n} w_i$;
$Q \leftarrow q_r$;
foreach $i \in [1, n]$ **do**
  $s_i \leftarrow 0$;
  $h_i \leftarrow \min(q_i, \frac{Qw_i}{K})$;
  $K \leftarrow K - w_i$;
  $Q \leftarrow Q - h_i$;

When Algorithm 1 ends the completion time of a task i is $\frac{w_i}{h_i}$. Moreover, it can be shown that $i < j \implies \frac{w_i}{h_i} \geq \frac{w_j}{h_j}$ (we omit that part of the proof).

Therefore, 1 has the latest completion time. There are two cases, either this completion time is $\frac{w_1}{q_1}$ (in which case it cannot be reduced) or it is $\frac{\sum_{i=1}^{n} w_i}{q_r}$ (in which case the resource is fully used). □

Theorem 2. *energetic_cumulative*$\backslash\{\underline{h}, x\}$ *is NP-complete for any* $x \in \{r, d, \overline{h}\}$

Proof. Membership: checking the energy equation is trivial and checking the resource equation is not more difficult than for the `cumulative` constraint.

We first prove hardness of `energetic_cumulative`$\backslash\{\underline{h}d\}$ by reduction π from an instance $x = \{x_1, \ldots, x_n\}$ of the SubsetSum problem of deciding the following proposition: $\exists I \mid \sum_{i \in I} x_i = s$.

Let $K = \sum_{i=1}^{n} x_i$. For every $i \in [1, n]$ we create a task i with $w_i = x_i, q_i = 1$ and $r_i = 0$. Then, we create three tasks L, M and R with $q_L = w_L = 2K - s, q_R = w_R = s + K$, $q_M = w_M = 2K$ and $r_L = 0, r_M = 1, r_R = 2$. Finally, we set $q = 2K$ and we ask if all the tasks can be scheduled with $H \leq 3$.

We show first that a solution of x entails a solution of $\pi(x)$. We set $s_L = 0$, $s_M = 1$, $s_R = 2$, $h_L = w_L$, $h_M = w_M$ and $h_R = w_R$. Now, let I be the subset of $\{1, \ldots, n\}$ such that $\sum_{i \in I} x_i = s$. For all $i \in I$, we set $s_i = 0$ and for all $i \in \{1, \ldots, n\} \setminus I$ we set $s_i = 2$ and for all $v \in \{1, \ldots, n\}$ we set $h_i = 1$.

Now suppose that there is a solution of $\pi(x)$. Observe first that in any solution, we have $s_R = 2$ and $h_R = w_R$ because $s_R \geq 2$ and $h_R \leq w_R$ and

any solution such that $s_R > 2$ or $h_R < w_R$ implies $e_R > 3$. Now, suppose that $h_M < w_M$, since $e_R \leq 3$ then M and R must overlap, and therefore, $h_M \leq q - w_R = 2K - (s + K) = K - s$. So we have $p_M = \frac{2K}{K-s} > 1$. By contradiction, we have $h_M = w_M$ and therefore, $s_M = 1$. By a similar reasoning we have $s_R = 0$ and $h_R = w_R$. The remaining tasks are therefore distributed so that they exactly fill two disjoint areas of size s and $K - s$. The sum of the energies of the tasks in the leftmost area is therefore equal to s.

The proof is the same for `energetic_cumulative`$\backslash\{\underline{h}r\}$ except that the constraints $r_L = 0, r_M = 1$ and $r_R = 2$ are replaced by $d_L = 1, d_M = 2$ and $d_R = 3$, and for every other task i, we have $d_i = 3$.

The proof is the same for `energetic_cumulative`$\backslash\{\underline{h}\overline{h}\}$ except that the constraints $\forall i,\ q_i = 1$ are relaxed, the constraints $q_L = w_L, q_R = w_R$, $q_M = w_M$ are replaced by $d_L = 1, d_M = 2$ and $d_R = 3$, and for $1 \leq i \leq n$, we have $d_i = 3$. □

5 Flow global constraints

Note that a global constraint similar to `energetic_cumulative`, called the continuous energy constraint, has been proposed in [9]. In this variant the rate of the activity is no longer constant and may vary over time. A flow-based propagator proposed for this constraint can be adapted for the `energetic_cumulative`. The propagator works by building a flow network relaxation $f(\mathcal{D})$ of the problem, before it propagates, using the current domain $\mathcal{D}$.

Given current domain $\mathcal{D}$, let $T_J = \{\underline{s}_i, \underline{e}_i, \overline{s}_i, \overline{e}_i \mid i \in J\}$ be the set of $O(|J|)$ minimum or maximum start and completion times of the tasks of J and let t_q be the q-th largest element of T_J. We partition the time line into the set of consecutive intervals $\mathcal{I}(J) = \{[t_q, t_{q+1}] \mid 1 \leq q < |T_J|\}$

Next, we create a flow $f(\mathcal{D})$ network as follows. We have source node S and a first layer of task nodes J, the flow from S to i is w_i for each $i \in J$. We create a second layer of time interval nodes $I_q = [t_q, t_{q+1}], 1 \leq q \leq |J| - 1$. There is an edge from i to I_q if $[t_q, t_{q+1}] \subseteq [\underline{s}_i, \overline{e}_i]$. The edge is bounded by $0..(t_{q+1} - t_q) \times \overline{h}_i$. If, in addition, $[t_q, t_{q+1}] \subseteq [\overline{s}_i, \underline{e}_i]$, the lower bound of the edge can be increased to $\underline{h}_i$. We create a final layer to a sink node E. There is an edge from each interval node I_q to E with capacity bounded by $(t_{q+1} - t_q) \times q_r$.

Theorem 3. *Any solution to* `energetic_cumulative`$((s_i, e_i, h_i, w_i)_{i \in J}, q_r)$ *given current domain $\mathcal{D}$, is extendible to a solution of the flow network $f(\mathcal{D})$.*

Proof. Given a solution of the constraint we extend it to a solution of the flow network $f(\mathcal{D})$ as follows. The flow from S to each node i is set to w_i. The flow from each node i to I_q is set to $(\min(e_i, t_{q+1}) - \max(s_i, t_q)) \times h_i$ The flow from each node I_q to E is given by the sum of the incoming flows to I_q. We show that these flows obey the bounds and flow balance equations. Examining each node i, the flow in is w_i and the flow out is $h_i \times (e_i - s_i)$. These must be equal by equation (5). Examining each node I_q the flow in is $\sum_{i \in \mathcal{E}} (\min(e_i, t_{q+1}) - \max(s_i, t_q)) \times h_i$ but by equation (4) at no time is there more than q_r resource being used, hence

this is no more than $(t_{q+1} - t_q) \times q_r$, the capacity of the outgoing arc. The flow balance at I_q holds by construction. □

Example 2. If we consider the circumstances explained in Example 1, the constraint `energetic_cumulative` generates the flow network shown in Figure 2.

While the standard propagator can determine nothing, the flow network is infeasible, so the `energetic_cumulative` propagator immediately fails.

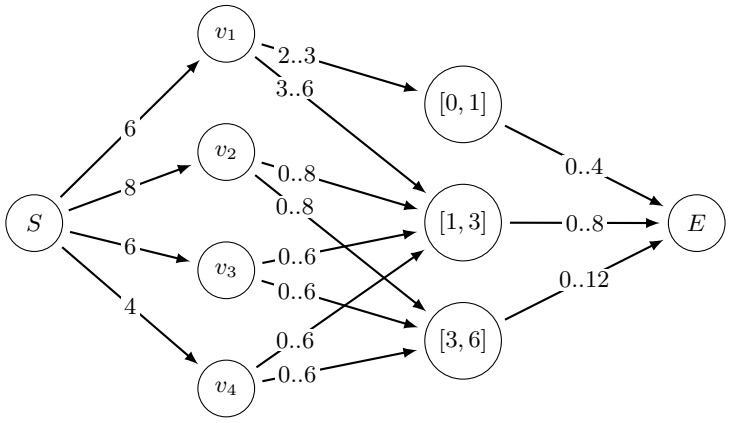


Fig. 2: Flow network for the tasks given in Table 1.

The natural way to implement this propagation for the evacuation planning problem at hand is to define one `energetic_cumulative` constraint per critical transit node $u \in \tilde{\mathcal{T}}$. The special structure of the evacuation planning problem allows us to consider simultaneously all `energetic_cumulative` constraints and to integrate the flow propagators of each critical transit node into a single global flow propagator. The basic remark is that all vehicles passing in an interval $[t_q, t_{q+1}]$ on a transit node u of an evacuation path will appear on an ascendant transit node $u' \in \hat{p}(u)$ exactly $l_{uu'}$ time units later, i.e. in interval $[t_q + l_{uu'}, t_{q+1} + l_{uu'}]$. Considering the evacuation tree reduced to the evacuation nodes and the critical transit node, the global network flow $gf(\mathcal{D})$ is built as follows. We create the source S and one layer of nodes for the evacuation nodes $\mathcal{E}$, the flow from S to $v \in \mathcal{E}$ being w_v. Let $T_v = \{\underline{s}_v, \underline{e}_v, \overline{s}_v, \overline{e}_v\}$ the time events associated to evacuation node $v \in \mathcal{E}$ and let $\mathcal{I}(v) = \{[t_q^v, t_{q+1}^v] \mid 1 \leq q < |T_v|\}$ the corresponding consecutive intervals. We connect node v to each interval $I_q^v \in \mathcal{I}(v)$ by an edge of capacity bounded by $0..\overline{h}_v \times (t_{q+1}^v - t_q^v)$ (or with a lower bound of $\underline{h}_v$ under the conditions described above). We assume that the critical nodes are sorted in the topological order from direct successors of the evacuation nodes to the direct predecessors of the safe nodes. Then, for each critical node $u \in \tilde{\mathcal{T}}$ taken in this order, we build recursively the set of events T_u from the events of its parent nodes in the evacuation tree and the corresponding consecutive interval set $\mathcal{I}(u)$: $T_u = \cup_{u' \in p(u)} \{t_q^{u'} + l_{uu'} \mid 1 \leq q < |T_{u'}|\}$ and $\mathcal{I}(u) = \{[t_q^u, t_{q+1}^u] \mid 1 \leq q < |T_u|\}$. We add an edge between each interval of

the parent node $I^{u'}_{q'} = [t^{u'}_{q'}, t^{u'}_{q'+1}] \in \mathcal{I}(u')$ and each interval of the child node $I^u_q = [t^u_q, t^u_{q+1}] \in \mathcal{I}(u)$ such that $t^{u'}_{q'} + l_{uu'} \leq t^u_q$ and $t^{u'}_{q'+1} + l_{uu'} \geq t^u_{q+1}$ with a capacity bounded by $0..(t^u_{q+1} - t^u_q) \times q_u$. Finally, we build an edge of unlimited capacity from each interval of each parent node $u \in p(r)$ of the safe node to the target E. The global flow has $O(|\mathcal{E}| + \sum_{u \in \tilde{\mathcal{T}}} L(u))$ nodes. The following theorem shows that if there is no feasible global flow, then `energetic_cumulative` fails.

Theorem 4. *Any solution to the conjunction of* $\texttt{energetic\_cumulative}((s_{i_{uv}}, e_{i_{uv}}, h_{i_{uv}}, w_v)_{v \in L(u)}, q_u)$ *for all* $u \in \tilde{\mathcal{T}}$ *given current domain* $\mathcal{D}$*, is extendible to a solution of the flow network* $gf(\mathcal{D})$*.*

Proof. The proof uses similar arguments as in the proof of Theorem 3. □

6 Heuristic upper bound

We propose a simple compression heuristic to find an initial upper bound. Assume w.l.o.g. that the evacuation rate initial domain is $\mathcal{D}(h_v) = [1, q_v]$ for all evacuation node The heuristic is based on the assumption that scheduling all evacuation tasks at time 0 with the minimum evacuation rate yields a feasible solution, with a high cost. Starting from this solution ($\forall v \in \mathcal{E}$, $s_v := 0$, $e_v = w_v$, $h_v := 1$), the set of minimum and maximum start and end time events T_u is built for each transit node, as well as the corresponding consecutive intervals $\mathcal{I}(u)$ as for the global flow constraint. In addition, the resource consumption profile Γ_u where $\rho_{qu} \in \Gamma_u$ denotes the resource consumption level right after time t^u_q is also computed for each transit node u.

At each iteration, the heuristic finds the critical task v^*, i.e. the one that sets the objective, as well as the second most critical task v'. The principle is to left shift the end of the critical task enough so that v' becomes the new critical task with an absolute gap of ϵ below the new cost of v^*. This yields a target completion time $e^* := \frac{c_{v'} - \epsilon}{w_{v*}} - d_{v*}$ for v^*.

However, a left shift of e_{v^*} means an increase of its height. On a transit node u, the evacuation task v ends at time $e_{v^*} + l_{uv^*}$. Let $I^u_{q^+}$ denote the interval starting with the evacuation completion time on u: $t^u_{q^+} = e_{v^*} + l_{uv^*}$. Let $I^u_{q^-}$ denote the interval starting with the evacuation start time on u: $t^u_{q^-} = s_{v^*} + l_{uv^*}$. The maximum height Δ_h increase is equal to $\Delta_h = \min_{q=q^-,\cdots,q^+-1}(q_q - \rho_{qu})$. Left shifting the completion time of the critical task v^* on u to the beginning of the preceding interval $t^u_{q^+-1}$ is possible if $\frac{w_w}{t^u_{q^+-1} - s_{v^*} - l_{uv^*}} \leq h_{v^*} + \Delta_h$. Otherwise, the critical task v^* can only be shifted to $s_{v^*} + l_{uv^*} + \frac{w_w}{h_{v^*} + \Delta_h} \in]t^u_{q^+-1}, t^u_{q^+}]$ and the maximum left shifted end time on transit node u has been found. In the first case, the process can be iterated by tentatively ending the task in interval $t^u_{q^+-2}$ till the maximum left shifted end time on u is found. Repeating this check on each resource yields a minimum possible left shifted completion time e. If $e \leq e^*$, then the left shift to e^* is possible and v' becomes the new critical task. The compression process restarts with $v^* := v'$. Otherwise, the left shift is only performed to e. v^* is still critical and the heuristic stops. Due to the possibility of only left shifting a task by ϵ at each iteration, the heuristic is pseudo polynomial.

7 Generating a realistic data set

Catastrophic wildfire requiring large population evacuation are, thankfully, rare events. However, it means that obtaining useful data is difficult, and indeed, this is a key problem within the GEO-SAFE project. A significant part of the project revolves around simulation tools such as EXODUS [6], however, even simulated data was hard to come by.

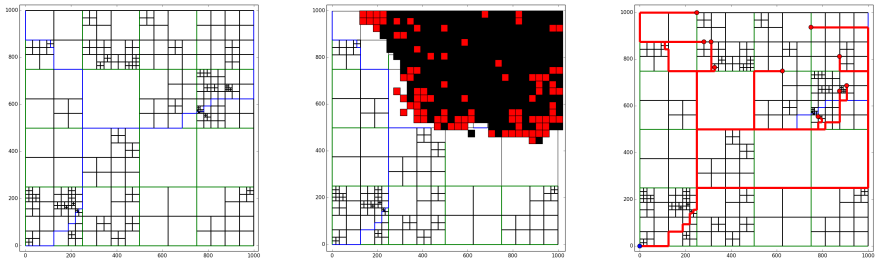
Therefore, we opted for taking advantage of the project environment to contribute to this effort by generating our own "realistic" dataset. On the one hand, this approach may introduce biases since we must use models to generate realistic road networks and simulate wildfires. On the other hand, we believe that it will make it much more convenient for benchmarking algorithms in the future. As it turns out, the generated instances are challenging even though relatively modest in size, thus being interesting from an academic viewpoint as well.

7.1 Generation of road networks

The first step is to generate a graph standing for the road network. To this end, we used the *quadtree* model described in [3]. In a nutshell, this model starts with a single square formed by four nodes and four edges. At each iteration, a square is chosen and five nodes are added, one in the center of the square, and one on each edge connected by a perpendicular edge to the center node. A parameter r controls the *sprawl*, that is, the preference for splitting larger squares ($r < 1$) or smaller squares ($r > 1$). The graphs generated in this manner share many features with real road networks: they are planar, embedded in an Euclidean plane, have similar density distributions, path lengths are within a constant factor of the Euclidean distance, and the number of turns is logarithmic with high probability. An example of random quadtree network is shown in Figure 3a. The colors on the edges correspond to road capacity. To allocate capacities, we first compute a minimum Steiner tree spanning three randomly chosen nodes in high density areas ("cities") and connect these cities to the nearest corner of the outer square. The corresponding set of edges are given the highest capacity and are coloured in blue in Figure 3a. A second set of edges, forming a grid are given an intermediate capacity, they are coloured in green.

7.2 Simulating wildfire

The second step consists in determining *safety due dates* for every edge of the evacuation tree, that is, a time after which the edge become unsafe. To this purpose we use a relatively simple fire propagation model. We chose to use a simple model based on two parameters: a constant *intensity* γ representing the type of fuel material as well as the temperature, and a *wind direction*. Indeed, the goal is not so much to accurately predict fire propagation, as it is to generate safety due dates consistent with a wildfire. Of course, should the authorities use



(a) road network (b) simulated wildfire (c) evacuation plan

Fig. 3: An example of generated instance

this type of planning tools during a real event, then correctly predicting fire propagation would be among the most important factors.

The land area is discretized into squares of fixed size (we use another parameter to control this size) which can be in three states: *untouched*, *burning* and *burned*. The fire starts as a single burning square, then at each iteration, any untouched square adjacent of a burning square catches fire with probability $\gamma(\frac{\pi-A}{\pi})^2$, where γ stands for the intensity of the fire, and A is the angle between the wind and a vector going from the center of burning square to the center of the untouched square. Moreover, any burning square that did not propagate stop burning with probability γ^2. Figure 3b illustrate the state of the simulated wildfire, with burning squares in red and burned squares in black.

7.3 Generating evacuation plans

The third step consists in generating the actual evacuation plan, that is, an embedded tree connecting a set of evacuation nodes $\mathcal{E}$ to a safe node r. Here again, the goal is not to compute the best evacuation plans, however, they must be representative of what would be actual plans.

We first randomly pick a predefined number of evacuation nodes among the nodes of the graph that are in the state *burned* or *burning* of the simulated fire. Then we use the convention that the safe zone is the furthest corner from the center of the fire. The evacuation tree is computed simply by using a shortest paths algorithm, however, with respect to an arc labeling taking into account first the safety due date of the arc, and only then its length and its capacity.

At this point we have all the information we need to define a fire evacuation problem as defined in Section 2. However, we use Observation 1 to remove redundant arcs. For every evacuation path, we explore the arcs from the safe node r to an evacuation node v. For any section of the path on which nodes have single child, all the nodes of that section have the same set of descendant leaves in the tree. Therefore, the same set of evacuees will go through these nodes with

the same relative delay. It follows that we conserve only the arc of minimum capacity in that section, and only if this capacity is strictly smaller than that of the previous section. It follows that an instance with n evacuation tasks has at most $O(n)$ "jam" constraints. Moreover, the objective can be stated as the maximum of $O(n)$ expressions as shown in Section 2.

The tools we developed as well as the benchmarks instances we used in this paper can be accessed at anonymized for the blind review process.

8 Experimental results

We generated 240 benchmark instances following the protocol described in Section 7. They are organized into three types of road networks: `Dense`, `Medium` and `Sparse` where the density refers to the number of intersections (respectively 400, 800 and 1200) in the land area. Notice that the graph has always 4 edges per node, so this corresponds to graph size. The impact on the instance is that larger graphs allow more choices for the shortest paths and therefore, longer independent paths. For every type of road network, we generated 4 classes of instances, with respectively 10, 15, 20 and 25 evacuation nodes. Finally, for every class we simulated 20 random wildfires and the subsequent evacuation trees. We used CPLEX to solve the flow formulation, which turned out to be extremely costly. Therefore, we did not systematically call the propagator at each node. Instead we implemented a heuristic method to decide whether we should solve the flow using two criteria:

The first criterion that we use is the total load $\Omega(r)$ of a resource r or capacity q_r on the tasks J:

$$\Omega(r) = \frac{\sum_{i \in J} w_i}{q_r(\max_{i \in J} e_i - \min_{i \in J} s_i)}$$

We only call the flow when $\max_{u \in \tilde{\mathcal{T}}} \Omega(u) \geq 0.95$, this value was empirically chosen. Moreover, observe that when $\Omega(r) > 1$ the constraint is trivially infeasible. In this case, we can return an early failure.

The second criterion is the size of the remaining search space. Indeed, solving the flow might not be worthwhile even it is infeasible, if brute-force search would have been faster. Here we empirically chose 2^{50} as a minimal search space, defined as the product of the ranges of all start time and height variables. That is, we solve the flow only when: $\sum_{i \in \mathcal{E}} \log |D(s_i)| + \log |D(h_i)| \geq 50$.

However, even with this approach, using the global flow constraint made CP Optimizer about one order of magnitude slower (measured by number of fails per second). Therefore, we also tested a version that *never* solves the flow, and only fails when the overall load is too large for the resource's capacity ($\Omega(r) > 1$).

We ran every method on every instance of the dataset with a time limit of 45 minutes on 4 cluster nodes, each with 35 Intel Xeon CPU E5-2695 v4 2.10GHz cores running Linux Ubuntu 16.04.4.

Table 2 shows results of running CP Optimizer with the strategy *Depth first search* (`DFS`). We use '^{h}' to denote the use of the heuristic upper bound described in Section 6, '$^+$' to denote the use of the overall load check described above, and

Table 2: Depth first search: upper bound and optimality ratio

		DFS		DFS^h		DFS^+		DFS^{h+}		DFS^f		DFS^{hf}	
		#s	ub	#s	ub	#s	ub	#s	ub	#s	ub	#s	ub
dense_10	(20)	1.00	26469	1.00	26469	1.00	25485	1.00	25573	1.00	25485	1.00	25573
dense_15	(20)	1.00	179099	1.00	176391	1.00	189864	1.00	188238	1.00	194737	1.00	195631
dense_20	(20)	1.00	365596	1.00	367030	1.00	406828	1.00	407116	1.00	486971	1.00	447924
dense_25	(20)	0.95	697661	1.00	663314	0.85	1093784	1.00	759437	0.85	1201755	1.00	881494
medium_10	(20)	1.00	49294	1.00	49294	1.00	49567	1.00	49567	1.00	49567	1.00	49567
medium_15	(20)	1.00	148513	1.00	143999	1.00	153372	1.00	149243	1.00	169840	1.00	170785
medium_20	(20)	0.95	339103	1.00	339370	1.00	376395	1.00	384381	1.00	507420	1.00	427572
medium_25	(20)	0.90	1291811	1.00	659269	0.80	1461162	1.00	722167	0.75	1286011	1.00	790511
sparse_10	(20)	1.00	4474	1.00	4759	1.00	4464	1.00	5951	1.00	4464	1.00	5951
sparse_15	(20)	1.00	131229	1.00	129276	1.00	135192	1.00	134420	1.00	146487	1.00	146917
sparse_20	(20)	0.90	293930	1.00	312834	1.00	320080	1.00	323449	1.00	388194	1.00	364291
sparse_25	(20)	0.95	598501	1.00	574534	0.80	593962	1.00	732975	0.80	684041	1.00	767851
avg	(20)	0.97	333506	1.00	287212	0.95	369874	1.00	323543	0.95	395298	1.00	356172

'f' for the flow propagator. The columns "#s" give the ratio of instances for which the method found a feasible solution, the columns "ub" the mean upper bound, and we highlight the best outcomes using colors. We can see that DFS does not find a feasible solution in every case, whereas our upper bound heuristic always finds one. Moreover, starting from a good quality solution pays off and the best solutions are significantly better than that of the baseline method. However, the flow propagator (DFS^f) and even the simple load checking rule (DFS^+) do not help and in fact decrease the performance of CP Optimizer.

Table 3: Default search: upper bound and optimality ratio

		CPO		CPO^h		CPO^+		CPO^{h+}		CPO^f		CPO^{hf}	
		ub	opt	ub	opt	ub	opt	ub	opt	ub	opt	ub	opt
dense_10	(20)	23294	0.95	23567	0.80	23354	0.95	23681	0.85	23503	0.90	23807	0.80
dense_15	(20)	161100	0.70	161700	0.75	162856	0.60	162751	0.55	164524	0.45	165025	0.45
dense_20	(20)	311676	0.45	312501	0.45	312697	0.15	313114	0.15	317035	0.10	316596	0.00
dense_25	(20)	531016	0.00	529889	0.00	530749	0.00	530975	0.00	539192	0.00	564722	0.00
medium_10	(20)	48591	1.00	48591	0.80	48591	0.95	48591	0.75	48776	0.90	48979	0.70
medium_15	(20)	124921	0.70	125195	0.50	125559	0.60	125110	0.60	126011	0.40	127421	0.40
medium_20	(20)	276101	0.25	276094	0.25	277545	0.10	277485	0.10	279654	0.05	278723	0.10
medium_25	(20)	488282	0.10	488555	0.10	491794	0.00	491423	0.00	493682	0.00	507275	0.00
sparse_10	(20)	3747	1.00	3880	0.90	3747	1.00	3880	0.90	3946	0.85	3880	0.85
sparse_15	(20)	120173	0.65	120151	0.65	120162	0.60	119803	0.55	120895	0.55	120576	0.55
sparse_20	(20)	235771	0.35	253807	0.35	237503	0.15	236920	0.20	238300	0.20	240112	0.15
sparse_25	(20)	437618	0.00	437219	0.05	439699	0.00	438160	0.00	441707	0.00	448979	0.00
avg	(240)	230191	0.51	231762	0.47	231188	0.42	230991	0.39	233102	0.37	237175	0.33

In fact, the results are even more negative when using CP Optimizer's default strategy. Table 3 shows the results in this setting, using the same conventions. The number of feasible solution is here replaced by the number of optimality proofs, since a feasible solution was found in every case. Here we can see that again the extra propagation does not improve the results. Moreover, the impact

Table 4: Ratio of failures due to the propagator

method	#sol	#nodes (total)	#fails (flow)	fails/node	speed
		avg	avg	avg	avg
CPO^f	1.00	525793	2371	0.01161	5324
CPO^+	1.00	3042714	2376	0.00137	30157
CPO	1.00	3084572	0	0.00000	34508

of the heuristic upper bound is milder: it helps finding better solutions in some cases, but overall it is actually slightly worse than not using it.

We find the empirical results puzzling. In the case of the flow propagator, we expect a tradeoff, not necessarily positive, between the extra pruning and the computational overhead. However, it is much more surprising that the load checker and the heuristic, which are both almost computationally free do not help at all, let alone *hinders* CP Optimizer. Table 4 shows the average number of nodes required to solve an instance, the average number of fails due to the flow, the ratio of fails per node and the "speed" in number of branches explored per second. On the one hand, we observe that the flow propagator decreases the speed by a factor 6, while failing in only about 1% of the calls. Recall, however, that the flow is solved only when the remaining search space is large enough, so these fails are likely to cut a relatively large subtree. On the other hand, we can see that the impact of the load checker on the search speed is rather limited, although visible, and that is does causes early failures (though admittedly rarely).

9 Conclusion

We have introduced a variant of the flood evacuation planning problem, where the objective function is adapted to the context of wildfires. We have designed a generator of challenging and realistic instances for this problem. We have introduced a new type of `cumulative` constraint where tasks have flexible duration and height, but a constant energy. We have analyzed the complexity of this constraint, and proposed several solution approaches, including an effective greedy upper bound and a global propagation method.

However, the experimental results we carried out suggests that these techniques tend to degrade CP Optimizer's performances. One possible insight that we can draw from these experiments is that the amount of tuning, sophisticated techniques, and the overall black-box nature of CP Optimizer, although making it extremely efficient in practice is also a problem when trying to test preliminary ideas on how to best tackle a problem. In order to improve over the results of CP Optimizer, it is therefore probable that we will need to use more refined algorithms and in particular dedicated flow algorithms which would be incremental and from which we could propagate bounds.

Acknowledgement

This work is partially funded by the H2020-MSCA-RISE-2015 European project GEO-SAFE (id 691161)

References

1. Hal E. Anderson. Aids to determining fuel models for estimating fire behavior. Technical Report 122, United States Department of Agriculture- Forest Service, Intermountain Forest and Range Experiment Station Ogden, UT 84401, April 1982.
2. Vedat Bayram. Optimization models for large scale network evacuation planning and management: A literature review. *Surveys in Operations Research and Management Science*, 2016. DOI: 10.1016/j.sorms.2016.11.001.
3. David Eisenstat. Random road networks: the quadtree model. *CoRR*, abs/1008.4916, 2010.
4. Caroline Even, Victor Pillac, and Pascal Van Hentenryck. Convergent plans for large-scale evacuations. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1121–1127, 2015.
5. Caroline Even, Andreas Schutt, and Pascal Van Hentenryck. A constraint programming approach for non-preemptive evacuation scheduling. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, pages 574–591, 2015.
6. Edward R. Galea, Mathew Owen, and Peter J. Lawrence. The EXODUS Model. *Fire Engineers Journal*, pages 26–30, 1996.
7. Geo-safe - geospatial based environment for optimisation systems addressing fire emergencies, MSCA-RISE-2015 - Marie Skłodowska-Curie Research and Innovation Staff Exchange (RISE) European projet – id 691161. http://fseg.gre.ac.uk/fire/geo-safe.html. Accessed: July 8, 2018.
8. Kanal Kumar, Julia Romanski, and Pascal Van Hentenryck. Optimizing infrastructure enhancements for evacuation planning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3864–3870, 2016.
9. Margaux Nattaf, Christian Artigues, and Pierre Lopez. Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions. *Constraints*, 22(4):530–547, 2017.
10. Victor Pillac, Manuel Cebrián, and Pascal Van Hentenryck. A column-generation approach for joint mobilization and evacuation planning. *Constraints*, 20(3):285–303, 2015.
11. Victor Pillac, Caroline Even, and Pascal Van Hentenryck. A conflict-based path-generation heuristic for evacuation planning. *Transportation research part B*, (83):136–150, 2016.
12. Richard C. Rothermel. A mathematical model for fire spread predictions in wildland fuels. Technical Report 115, USDA For. Serv., lntermt. For. and Range Exp. Stn., Ogden, Utah, USA, 1972.
13. Shahrooz Shahparvari. *Enhancing Emergency Response in Short-notice Bushfire Evacuation*. PhD thesis, RMIT University, 2016.

14. Shahrooz Shahparvari, Prem Chhetri, Babak Abbasi, and Ahmad Abareshi. Enhancing emergency evacuation response of late evacuees: Revisiting the case of australian black saturday bushfire. *Transportation Research Part E: Logistics and Transportation Review*, 93:148 – 176, 2016.
15. Alexander Stepanov and James MacGregor Smith. Modeling wildfire propagation with delaunay triangulation and shortest path algorithms. *European Journal of Operational Research*, 218(3):775 – 788, 2012.
16. Anand Veeraswamy, Edwin R Galea, Lazaros Filippidis, Peter J Lawrence, and Robert J Gazzard. The simulation of urban-scale evacuation scenarios: Swinley forest fire. In *Proceedings 6th Int Symp Human Behaviour in Fire*, pages 221–232, 2015.
17. David R. Weise and Gregory S. Biging. A qualitative comparison of fire spread models incorporating wind and slope effects. *Forest Science*, 43(2):170–180, 1997.