The Explicit Decision Algorithm for Undergrounding or Hardening Power Lines in Fire-Prone Communities

RTSync Corp. — October 2026

Simulation study: why narrow, windy hillside streets and complex topography require undergrounding utility lines to avoid cascading evacuation delays.

Executive Summary

RTSync Corp. — October 2026

The situation that this submission addresses

We live in a hillside community characterized by a Dendritic street pattern (or Tree-Like) network where minor local streets feed into collector streets, which then funnel traffic into a small number of arterial exit points. During a wind-driven fire, power lines fall across streets and a downed line on a narrow street is repaired only as fast as the crew can get through the cars it has trapped; a second line further up turns that delay into a cascade the crew may never catch up with. Our simulation model concludes that this scenario could lead to catastrophic consequences that would be totally mitigated by undergrounding of powerlines.

Under intense wind, power lines that fall across streets do two things at once. They close the street, and they fill it: every car that was heading out along that street stops and waits. The utility's crew that must make the line safe before traffic can pass has to reach it by driving *into* the neighbourhood, against the flow of people leaving, through the very queue the fallen line created. On a wide road with a free inbound lane this costs little. On a single-lane hillside street with cars parked on both sides, each queued vehicle has to manoeuvre to let the truck by, and the crew's progress is set by the queue, not by the distance.

Current practice treats "time to repair" as a property of the span of line: thirty minutes, an hour, two hours, from utility records. This summary reports what happens when that assumption is replaced by a crew that must physically reach the site, and what follows when there is more than one site.

What was done

RTSync undertook a simulation study to elucidate the factors that impact the decision to require undergrounding utility lines to avoid cascading evacuation delays. The simulation model was constructed to replicate how repair of downed lines might proceed in both grid and

dendritic street patterns. This scenario considers a sequence where 1) one repair crew leaves the main exit the moment the lines fall, 2) crosses each street at its free-flow time plus the time to pass every vehicle queued on it, 3) repairs the span, and then 4) moves to the next reachable downed span. The rate at which queued cars can be passed was considered a parameter and swept from the normal case (about two seconds per car) down to sixteen seconds per car and beyond, on the two community layouts: a grid of streets with several ways out, and a dendritic street pattern where each street is the only way out for the homes above it. One, two and three downed spans were examined, with the second placed behind the first on the same route. Residents were assumed, as a baseline, not to change what they do — they follow their usual route and wait where it is blocked, because any other behaviour requires information they may not have.

What was found

1. Up to a point, a slow crew costs nothing; past it, every minute of crew delay is a minute of evacuation delay. As long as the crew frees the trapped group before the community's exits would have finished anyway, the trapped residents simply join the end of the queue and the clearance time is unchanged. On the tree layout this held until the pass rate fell to about one car every twenty-four seconds (a crew taking two hours to drive a mile of queued street). Beyond that threshold, clearance rose one-for-one with crew travel time: 212, 251, 328 and 406 minutes against a normal 176, at pass rates of 24, 32, 48 and 64 seconds per car. The threshold is sharp and the slope above it is one; there is no gentle degradation.

2. A second site behind the first turns delay into a cascade. With one downed span the crew passes one queue. With a second span further up the same route, the crew passes the first queue, repairs the first span, and then passes the second queue — which has grown the whole time it was waiting. In the simulation the crew passed 290 vehicles for one site and 690 for two. The threshold at which crew access dominates drops from twenty-four to eight seconds per car, and at sixteen seconds per car and worse the community does not clear within ten hours. Each additional site adds not only its own repair time but the queue that formed behind it while the crew was elsewhere, and that queue is itself what slows the crew. This is the cascade: block → queue → slower access → longer block → longer queue.

3. The grid is no protection if nobody uses it. When residents on the grid layout wait at the block as they would on the tree, the same thresholds appear: crew access dominates from eight seconds per car for any number of sites. The alternative routes exist; they are simply unused.

4. Information changes the grid entirely and the tree hardly at all. The simulation was repeated under a weaker assumption: a driver who arrives at a blocked street sees the downed line, turns around, and takes another way if one exists; nobody else changes. On the grid this removed the entire crew-access penalty — clearance stayed at its normal value at every pass rate, because the trapped column found its own way out and the crew's lateness no longer mattered. On the tree it changed nothing for one or two downed spans, because there is no other way, and gained twenty-one minutes for three, where one blocked lane happened to lie near the community's little-used second exit. A fully

coordinated evacuation — an alert naming the blocked streets, or officers at junctions — did no better than local knowledge on either layout.

What this means

For the utility's choice between burying and hardening. The repair-time figure that enters any underground-versus-harden calculation must include access: the time to reach the span through the streets between it and the exit, given their width and the queue that will be on them. That makes it a property of the network, not of the span, and it is largest exactly where the fire risk is also largest — narrow, tree-shaped hillside streets. A hardened line on such a street is a line whose eventual fall cannot be fixed in time, whatever the nominal repair figure says. The same span on a wide collector with an inbound lane can be hardened with less concern. The current benefit-cost procedure cannot see this distinction; its qualitative "ingress/egress" flag gestures at it without a number.

For the cascade. Where several spans share one route — the usual case on a canyon road with lines along its whole length — the risk is not the sum of the individual spans' risks. Burying the span nearest the exit does not help if a span further up falls, and burying the one further up does not help if the crew cannot get past the first. Mitigation on a route is effective only when it covers the route; partial undergrounding of a tree-shaped street leaves the cascade intact. This is the quantitative reason the decision algorithm removes undergrounding span by span only while the whole community still clears in time, rather than scoring spans one at a time.

For emergency management. On a grid-shaped community, telling residents which streets are blocked is worth as much as burying the lines on those streets — the model shows the whole penalty disappearing. On a tree-shaped community, information cannot substitute for infrastructure: there is nothing to tell people except to wait. The practical consequence is a division of effort: alerting systems and traffic control for the grids, undergrounding for the trees. A community should know which kind it is, street by street, before either is proposed.

For the regulator. Three measurable quantities decide the outcome and none of them appears in the current filing: the pass rate on each street (lanes, parking, width), the number of downed spans on one route in a wind event, and whether residents will receive and act on information about blocked streets. The first two can be surveyed and modelled; the third is a plan that can be required and audited. An explicit decision rule can take all three as inputs; a benefit-cost ratio with a fixed repair time cannot.

Note the premise of utility crews repairing fallen lines during active evacuations is a representation of the failure mode: a fallen line should represent an **impassable hazard** until public safety agencies cut or bypass it, removing the speculative "crew pass rate" metric and replacing it with an "emergency obstacle clearance" distribution.

The below sweep of all configuration scenarios is done via the ParaDEVS branching functionals and is guaranteed to find a conservative lower bound on safe clearance probabilities. It is not a greedy monotone searcher needing a globally robust optimization

solver (such as branch-and-bound or mixed-integer programming with submodular relaxation) to guarantee that non-linear network interactions do not produce unsafe span selections.

Summary

A downed line on a narrow street is repaired only as fast as the crew can get through the cars it has trapped; a second line further up turns that delay into a cascade the crew may never catch up with; on a grid, telling residents where the congestion points are helps to ameliorate the situation -- but burying the lines on a dendritic street pattern is the only solution.

Introduction

Why An Explicit Decision Algorithm

Consider:

During the 1991 fire it took 3 hours for people to get out. Some people could not get out of their driveways due to the clogged streets and wound up running down the hill. Many residents did not evacuate.

1. **"It took three hours for people to get out."** Recounts of evacuees are consistent with the 3 hour estimate although no official report gives a single evacuation duration, and the figure does not appear in the NFPA, U.S. Fire Administration, or State OES reviews.. The OES multi-agency review states that "in some cases, fire apparatus and private vehicles were trapped in areas for several hours," and the NFPA report that vehicles were "trapped for hours on these roads." A Montclair resident's first-person account describes all four lanes of Mountain Boulevard running one-way outbound, bumper to bumper, "orderly but slow," with safety only reached after crossing Highway 13 at Park Boulevard. The timeline matters more than the duration: the fire flared at 10:45, blew out of the canyon at 11:15, reached the Parkwoods Apartments by 11:30 and Hiller Highlands shortly before noon; most deaths occurred between 11:30 and 12:00. For the people who died, the race was lost in 15 to 45 minutes, not three hours — the three-hour figure describes the survivors' experience on the roads below.

2. **"Some people could not get out of their driveways and ran down the hill."** Well supported. The NFPA report: traffic jams developed, residents "abandoned their cars in frustration so they could run to safety," and the abandoned cars then became roadblocks; "over half of the fatalities occurred in cars that burned on the winding roads." The USFA report: at Parkwoods (456 units, one gate), engines reported more than 1,000 residents trying to escape by 11:37, and "collisions and falling power lines blocked narrow streets." On Charing Cross Road a collision jammed cars in the narrowest section; eleven

people died there, trapped in the queue, and eight more "on narrow streets in the same area." Officer Grubensky died with the people he had loaded into his car when their route was blocked; Battalion Chief Riley was electrocuted by a falling line while helping a woman leave by car. In the OES tally of engine-company problems, downed power lines were the most common (185 reports), ahead of water supply (165) and narrow steep roads (132)

3. **"Many residents did not evacuate."** Supported, in three forms. Some refused: the OES review records residents "refusing to evacuate until the last possible moment," preparing hoses and sprinklers, and a woman on her rear deck resisting orders while the front of her house burned. Some returned: residents re-entered to check on people or property and had to be evacuated again. And many were never warned: the Emergency Broadcast System was not used by any agency; warning was door-to-door and by loudspeaker; Berkeley dispatchers told callers at 11:41 "it's Oakland's fire, they're taking care of it" and at 12:34 "there have been no evacuations, as far as I know"; NFPA concludes most fatalities involved people who "had little warning." Those who survived in place did so by sheltering — seven civilians and an engine crew in a concrete-block garage, a homeowner and two firefighters under a pool cover for over an hour.

Why it still matters. The roads have not changed: a 2011 retrospective quotes the Diablo Fire Safe Council that "the width of the narrow roads hasn't changed at all." Berkeley's 2025 KLD evacuation-time study — outbound traffic only, no road blockages, no inbound crews — found 4 h 10 min to clear the hills on a school-day afternoon, 1 h 35 min in the evening, and 45 min to 1 h 45 min for Panoramic Hill's single road; a Stanford reviewer called these "a likely minimum." Oakland has no equivalent published study.

The proposed decision algorithm and its simulation model support are consistent with these recounts. The record corroborates the mechanisms in the decision algorithm and suggests three calibration points: downed lines were the leading cause of blocked streets, not an edge case; the lethal failure mode was a queue stopped at a narrow point by a collision or a line, with the fire arriving within minutes; and the resident-behaviour assumption should allow for a substantial fraction who are not warned, refuse, or return (see References and Sources).

Utilities in California and other fire-prone regions must decide, span by span, whether to bury their overhead distribution lines or harden them in place with insulated conductor and stronger poles. The method now used to make and review that decision is a benefit-cost ratio: the expected dollar value of avoided fires and outages divided by the cost of the work. It has two defects that matter in a hillside community. First, because both options are credited with a fixed fraction of the same dollar risk, the risk cancels when they are compared, and the choice reduces to a ratio of construction costs; how dangerous the place is drops out. Second, the one thing residents of such a community fear most — a fallen line across the road out while the fire is coming — is absent from the arithmetic, and enters only as an unmeasured judgment made by utility field staff.

The algorithm described in this document replaces that judgment with a measurement and puts safety ahead of cost rather than inside it. It is explicit in the plain sense: a reader with the published inputs can rerun it and obtain the same answer.

The algorithm in outline

The algorithm treats a community as a single evacuation network, because a closed street changes the load on every other street. It works in six steps.

1. **Measure the race.** For each wind-and-fire scenario, fire-spread simulation gives the time the fire takes to reach the community; a traffic model of the community's streets and exits gives the time the community takes to empty. The difference is the egress margin. A negative margin means people are still inside when the fire arrives. This, not a dollar figure, is the quantity the decision protects.

2. **Let the mitigation change the margin.** Each span of line, under each option — bare, hardened, or underground — has a probability of falling across a street in a wind event and a distribution of repair times. A fallen span closes its street and traps whoever is upstream until a repair crew, driving against the evacuation and through the queue the fall created, reaches and clears it. Undergrounding removes the span from this calculation; hardening reduces the chance of a fall but not its consequences. The algorithm computes, exactly rather than by random sampling, the probability that the community clears in time under every combination of options on the spans that control its exits.

3. **Set the requirement.** The regulator fixes, once and for every community alike, the probability with which the community must be clear before the fire arrives and the safety buffer in minutes. An option set is admissible only if it meets that requirement within budget.

4. **Test feasibility first.** If even burying every controlling span cannot meet the requirement, the algorithm says so, names the scenarios and the households at risk, and stops: the community needs another exit or an evacuation plan, not a different line treatment.

5. **Find the cheapest admissible set.** Starting from full undergrounding, the algorithm removes undergrounding from one span at a time and keeps each removal only if the requirement still holds. Hardening is therefore chosen wherever the network can absorb that span's fall — common on a grid of streets with alternatives — and refused wherever the fall would strand households past the fire's arrival — common on a tree of hillside lanes. Cost-benefit analysis is applied last, among admissible sets only.

6. **Publish.** The inputs, the enumerated scenario outcomes, the search trace, and a statement of the form "this set of treatments clears the community before the fire with probability p at cost c; full undergrounding achieves p′ at c′; hardening alone achieves p″ at c″" go on the record.

Two refinements, both arising from simulation studies reported in the companion executive report, are built in. The repair time of a span is an *effective* time that includes the crew's access through queued, narrow streets, and is therefore a property of the network rather than of the span; on single-lane dendritic streets it can be the dominant term, and a

second fallen span behind the first turns delay into a cascade. And the behaviour assumed of residents at a blocked street — waiting, turning around on sight, or acting on an alert — is an explicit input, with the uninformed case as the default unless an alert or traffic-control plan is on record.

How this document is organised

The main text that follows states the algorithm in plain language with named quantities (`RequiredClearProbability`, `FireArrivalMinutes`, `CrewPassRate`, and so on), so that a council member, an emergency manager, or a regulator can follow every step without notation. It covers, in order: the inputs (§1), the basic concepts (§2), the evacuation model (§3), the resident-behaviour assumption (§4), how the probabilities are worked out (§5), the decision steps as pseudocode (§6), and a notional use case (§7).

Two appendices support the main text.

Appendix A — The Algorithm in Mathematical Form restates every section of the main text, one for one, in standard mathematical and computational notation: the inputs as sets and functions, the concepts as numbered definitions and equations (A.1)–(A.15), the evacuation model as a discrete-time system, the enumeration as pseudocode with its lower-bound property, and the decision as an optimisation with a stated admissible set. It closes with a table mapping every plain name in the main text to its symbol. It is written for reviewers who need to check the algorithm's logic or implement it independently.

Appendix B — Simulation Models and Experimental Frames describes the software with which the results in this document were produced: the DEVS evacuation-network model and its repair-crew extension, the ParaDEVS enumeration that yields exact outcome probabilities, and the EFSUT experimental frames that tie each regulatory question to a declared, reproducible experiment. It records the verification performed, the illustrative grid and tree networks used, and the parameter sweeps that established when crew access becomes the dominant factor and what information or coordination is worth. It is written for the modelling team and for any party wishing to reproduce or extend the results.

Readers who want only the conclusion should read §6 and §7 of the main text. Readers who want to audit it should read Appendix A. Readers who want to run it on a named community should read Appendix B and then supply the inputs of §1.

Appendix A The Explicit Decision Algorithm

RTSync Corp. — October 2026 — companion to the Full Specification (v1.0)

---

1. Input Parameters

Name	Meaning	Example
`RequiredClearProbability`	How often the community must be clear of vehicles before the fire arrives, over the	0.99 (99 of 100 events)

Name	Meaning	Example
	whole set of wind scenarios	
`SafetyBufferMinutes`	Minutes the community must be clear *before* fire arrival, to leave room for crews working the exits	30
`Budget`	Money available for this community's lines	$
`PlanningHorizonMinutes`	Vehicles still inside this long after the order are counted as trapped whatever happens later	600
`ProbabilityFloor`	Scenario branches less likely than this are not followed; their total probability is reported as unresolved	0.0001
`CostTolerance`	Among safe options, those within this fraction of the cheapest may be compared on cost-benefit	0.30

Community Configuration

- *`RoadNetwork`*: the intersections and the streets between them. Each street has `LengthMinutes` (free-flow travel time), `Lanes`, `StreetCapacity` (vehicles per hour it can pass), and `CrewPassRate` (vehicles per minute a repair truck can get past when the street is full of queued cars — high on a wide street with an inbound lane, low on a single lane with parking).
- *`Exits`*: the intersections where the community's streets join the regional roads; each has `ExitCapacity` (vehicles per hour).
- *`VehiclesAt`*`[intersection]` by time of day (at least a daytime and an evening value).
- *`NormalRoute`*`[intersection]`: the street each household takes to its exit under normal conditions.
- *`Spans`*: every stretch of overhead line, with its `SpanLength`, the streets it could fall across, and `HouseholdsStrandedIfDown`: the households whose only route uses a street this span crosses. A span with any such households is a **controlling span**.

Powerline Options

`Options` are `Bare`, `Hardened`, `Underground`.

- *`Cost`*`[span, option]`
- *`ProbSpanFalls`*`[span, option, scenario]`: chance the span comes down across the street in that wind scenario (zero for Underground)
- *`RepairWorkMinutes`*`[span, option]`: a short list of possible repair durations with their probabilities, from utility records, e.g. 30 min (40 %), 60 min (40 %), 120 min (20 %)

Representation of Fire and Weather

ScenarioSet: a declared list of scenarios. Each one fixes the wind class, the time of day, the ignition point, *FireArrivalMinutes* (from fire-spread simulation, measured from the evacuation order), the *ResidentBehaviour* to assume (see §4), and *ScenarioWeight* (how often this scenario occurs among wind events; the weights sum to one).

---

2. Basic Concepts

- **Configuration**: one choice of option for every controlling span. `ConfigurationCost` is the sum of its span costs.
- **BlockageState**: within one scenario, which spans actually fall and how long each repair takes. Its probability is the product of the per-span chances along the way.
- **ClearanceMinutes**: time from the evacuation order until the last vehicle is out, for one configuration, one scenario and one blockage state — computed by the evacuation model.
- **EgressMargin** = `FireArrivalMinutes − ClearanceMinutes`. Negative means people are still inside when the fire arrives.
- **ResidentsTrapped**: residents still inside at `FireArrivalMinutes`.
- **ProbClearInTime** [was P_ok]: the probability, over all scenarios and blockage states, that `EgressMargin ≥ SafetyBufferMinutes`.
- **Admissible** configuration: `ProbClearInTime ≥ RequiredClearProbability` and `ConfigurationCost ≤ Budget`.

---

3. The evacuation model

The basic evacuation model supports minute-by-minute simulation of the road network:

7. At the order, every household's vehicles start down `NormalRoute`.
8. Each street passes at most `StreetCapacity` per hour; the rest queue at its upstream end. Each exit passes at most `ExitCapacity`.
9. At `LineFallMinutes` (set in the scenario), the spans in the blockage state fall and close their streets.
10. **Repair crew.** One crew leaves the main exit when the lines fall. To reach a span it drives street by street against the outbound traffic; crossing a street takes `LengthMinutes` plus (vehicles queued on it) ÷ `CrewPassRate`. It repairs the span for `RepairWorkMinutes`, then goes to the nearest remaining downed span it can reach, and so on. So:

    ```
    EffectiveRepairMinutes[span] = CrewArrivalMinutes[span] +
    RepairWorkMinutes[span]
    ```

and `CrewArrivalMinutes` depends on the streets and queues between the exit and the span — a property of the network, not of the span.

11. Vehicles upstream of a closed street behave according to `ResidentBehaviour` (§4).
12. The run ends when the last vehicle is out or at `PlanningHorizonMinutes`.

The model is deterministic: the same configuration, scenario and blockage state always give the same `ClearanceMinutes`. All chance is handled by enumeration (§5).

---

4. Resident behaviour (an explicit assumption, not a hidden one)

`ResidentBehaviour`	What drivers do at a closed street	What it assumes
`Uninformed`	Follow the normal route; wait at the block until the crew clears it	Nothing. **Default for the decision.**
`LocalKnowledge`	A driver who reaches the block turns around and takes another way if one exists; nobody else changes	Drivers know their own neighbourhood
`Alerted`	Everyone re-plans the moment the block occurs	A working alert system or officers at junctions

The decision is made under `Uninformed` unless the community can show the alert or traffic-control plan that justifies a stronger assumption; the plan then becomes part of the record.

---

5. Working out the probabilities

For one configuration and one scenario, build the tree of blockage states:

```
EnumerateBlockages(configuration, scenario):
    start with "no span has fallen", probability 1
    for each controlling span in turn:
        branch A: the span does not fall -> multiply probability by (1 -
ProbSpanFalls)
        branch B, C, ...: it falls, repair takes one of the listed durations
              -> multiply by ProbSpanFalls × that duration's probability
        drop any branch whose probability is below ProbabilityFloor
            (add it to UnresolvedProbability)
          each surviving path is one BlockageState with its probability
```

Spans that strand nobody are skipped; spans that close the same street for the same time are merged. Run the evacuation model once per leaf; the probability masses of leaves with the same `ClearanceMinutes` are added together. The result is the exact list of possible clearance times with their probabilities for that configuration and scenario. Then to obtain the probability that all cars clear in time, ProbClearInTime , aggregate all the leaves of each scenario and weight the results by the scenario probabilities:

```
Let SafeLeaves be those for which EgressMargin ≥ SafetyBufferMinutes.
```

```
ProbClearInTime(configuration/scenario) = Σ over SafeLeaves in the scenario
of their probabilities.

ProbClearInTime(configuration) = Σ over scenarios of
ProbClearInTime(configuration/scenario) x Probability weight of scenario

Note that ProbClearInTime is a lower bound because some of the
UnresolvedProbability may be attributable to safe clearance but not pursued
in branching.
```

---

6. Decision Algorithm Pseudo Code

```
Step 1 — Can the requirement be met at all?
    FullUnderground = every controlling span set to Underground
    if ProbClearInTime(FullUnderground) < RequiredClearProbability:
        report INFEASIBLE: the scenarios where the margin is negative,
               ResidentsTrapped in each, and the parts of the community responsible
        stop  (the community needs another exit, a staged evacuation plan,
               or a different requirement — burying lines will not do it)

Step 2 — Does the budget reach?
    if ConfigurationCost(FullUnderground) > Budget:
        continue to Step 3; if nothing admissible is found,
        report BudgetShortfall = (cost of the cheapest configuration that meets
                                  the requirement) - Budget

Step 3 — Cheapest configuration that still meets the requirement
    if there are 12 or fewer controlling spans:
        try every configuration; keep those that are Admissible;
        Selected = the cheapest (ties: the higher ProbClearInTime)
    else:
        Current = FullUnderground
        repeat:
            for each span not yet Bare:
                Trial = Current with that span downgraded one step
                        (Underground -> Hardened, or Hardened -> Bare)
                if Trial is Admissible:
                    Value = (money saved) ÷ (loss in ProbClearInTime)
            if no Trial is Admissible: stop
            Current = the admissible Trial with the largest Value
        Selected = Current
    (Hardening is kept wherever the network can absorb that span falling;
     it is refused wherever the span's fall would strand households past
     the fire's arrival. On a grid of streets with unused alternatives this
     will be often; on a Dendritic pattern it will be rare.)

Step 4 — The statement that goes on the record
    for each of Selected, FullUnderground, AllHardened:
        ProbClearInTime, ProbNegativeMargin, the worst 1-in-20 margin,
        ∑ over the scenarios that fail and ResidentsTrapped in each,
        expected trapped-vehicle-minutes, ConfigurationCost


Step 5 — Cost-benefit, last
```

```
    Candidates = admissible configurations costing no more than
                 (1 + CostTolerance) × ConfigurationCost(Selected)
    Final = the candidate with the best benefit-cost ratio
            (the utility's own ratio: ignition and outage benefits ÷ cost)
    The ratio chooses among safe options; it never overrides admissibility.

Step 6 — Publish
    all inputs of §1, the leaf tables of §5 for the three configurations,
    the trace of Step 3, the statement of Step 4, and Final.
    Anyone with the same inputs must get the same Final.
```

---

7. Notional Use Case Example

A hillside community with two exits, one a Main Exit (900 vehicles/hour) and one a Narrow Exit (450/hour, single lane) with 4,220 vehicles that try to exit has three controlling span groups: `MainApproach` (on Main Exit's road), `NarrowLeg` (on the single-lane leg), and `UpperLane` (a dead-end lane serving 60 homes). Fire is projected to arrive in 240 minutes. The buffer 20; required probability 0.99. Normal clearance 206 minutes.

1. Burying everything: clears the community in time with probability 1.00, cost 3.0 units.
2. Burying MainApproach and NarrowLeg, hardening UpperLane: still 1.00 (the lane's 110 vehicles are out long before 240 even if it falls), cost 2.3.
3. Burying only MainApproach: probality 0.970 — NarrowLeg falling for two hours pushes clearance past the limit three times in a hundred. Rejected in Step 1.
4. Hardening everything: 0.88. Rejected in Step 1.

Statement: "Undergrounding the Main Exit and Narrow Exit approaches and hardening Upper Lane clears the community before a 240-minute fire with probability 1.00 at 77 % of the full-underground cost; hardening everything achieves 0.88."

Note that if we change the fire time to arrival to 360 minutes then hardening everything reaches probability of clearance of approximately 0.995 and becomes the selected alternative. On the other hand, consider an extreme Dendritic configuration of single-lane streets that all converge to a single exit with a 150-minute fire time to arrival. In this case only burying every controlling span passes.

Appendix A — The Explicit Decision Algorithm in Mathematical Form

RTSync Corp. — October 2026

This appendix restates the algorithm of the main text in standard mathematical and computational notation. Section numbers and item order correspond one-to-one with the main text (§1 Input Parameters … §7 Notional Use Case). A correspondence table of names and symbols is given in §A.8.

A.1 Input Parameters

A.1.1 Regulatory constants

Symbol	Main-text name	Domain
α	`RequiredClearProbability`	$\alpha \in (0,1]$
β	`SafetyBufferMinutes`	$\beta \geq 0$
B	`Budget`	$B \geq 0$
H	`PlanningHorizonMinutes`	$H > 0$
ε	`ProbabilityFloor`	$\varepsilon \in [0,1)$
τ	`CostTolerance`	$\tau \geq 0$

A.1.2 Community configuration

Road network $G = (V, E)$ with intersections V and streets $E \subseteq \{\{u, v\}: u, v \in V\}$. For each street $e \in E$:

- $\ell_e \geq 0$ — `LengthMinutes`, free-flow traversal time
- $\lambda_e \in \mathbb{N}$ — `Lanes`
- $c_e > 0$ — `StreetCapacity`, vehicles per hour
- $\rho_e > 0$ — `CrewPassRate`, vehicles per minute a repair vehicle passes when the street is queued

Exits $X \subseteq V$, each with capacity $c_x > 0$ (`ExitCapacity`), vehicles per hour.

Demand $n_v(\theta) \geq 0$ — `VehiclesAt[v]` at time of day $\theta \in \Theta$.

Normal routing $r: V \to V$, with $r(v)$ the next intersection on v's route to its exit (`NormalRoute`); $r(x) = x$ for $x \in X$. The route from v is the sequence $v, r(v), r\big(r(v)\big), \ldots$ terminating in X.

Spans S. For $s \in S$: length ℓ_s; crossed streets $E_s \subseteq E$; stranded set

$$K_s = \{\, v \in V : \text{every route from } v \text{ to } X \text{ uses some } e \in E_s \,\}$$

(`HouseholdsStrandedIfDown`). Controlling spans $S_c = \{s \in S: K_s \neq \emptyset\}$.

A.1.3 Powerline options

Option set $O = \{\text{bare}, \text{hard}, \text{ug}\}$. For $s \in S$, $o \in O$:

- $\text{cost}(s, o) \geq 0$ — `Cost[span, option]`
- $p_s(o, w) \in [0,1]$ — `ProbSpanFalls[span, option, scenario]`, with $p_s(\text{ug}, w) = 0$
- $f_s(\cdot \mid o)$ — `RepairWorkMinutes[span, option]`: a probability mass function on a finite set $R_s(o) \subset \mathbb{R}_{>0}$ of repair durations, $\sum_{r \in R_s(o)} f_s\,(r \mid o) = 1$

A.1.4 Fire and weather

Scenario set W. Each $w \in W$ is a tuple

$$w = \big(\text{wind class},\ \theta_w,\ \text{ignition point},\ T_{\text{fire}}(w),\ t_{\text{fall}}(w),\ b_w,\ \pi(w)\big)$$

with $T_{\text{fire}}(w) > 0$ (`FireArrivalMinutes`), $t_{\text{fall}}(w) \geq 0$ (`LineFallMinutes`), $b_w \in \{U, L, A\}$ (`ResidentBehaviour`, §A.4) and weights $\pi(w) > 0$ (`ScenarioWeight`), $\sum_{w \in W} \pi\,(w) = 1$.

A.2 Basic Concepts

Configuration. $\kappa: S_c \to O$ (`Configuration`). Cost (`ConfigurationCost`):

$$C(\kappa) = \sum_{s \in S_c} \text{cost}\big(s, \kappa(s)\big).$$

Blockage state. For scenario w and configuration κ, a blockage state is a pair $b = (F, \rho)$ with $F \subseteq S_c$ the fallen spans and $\rho: F \to \mathbb{R}_{>0}$, $\rho(s) \in R_s\big(\kappa(s)\big)$, the repair durations. Its probability is

$$P(b \mid \kappa, w) = \prod_{s \in F} p_s\,(\kappa(s), w)\, f_s\big(\rho(s) \mid \kappa(s)\big) \prod_{s \in S_c \setminus F} \big(1 - p_s(\kappa(s), w)\big).$$

Clearance time. $T_{\text{clear}}(\kappa, w, b) \in (0, H]$ (`ClearanceMinutes`), the output of the evacuation model of §A.3.

Egress margin.

$$M(\kappa, w, b) = T_{\text{fire}}(w) - T_{\text{clear}}(\kappa, w, b).$$

Residents trapped. $N_{\text{trap}}(\kappa, w, b) = \sum_{v \in V} \big(\text{vehicles of origin } v \text{ still inside at } t = T_{\text{fire}}(w)\big) \times$ (residents per vehicle) (`ResidentsTrapped`).

Probability of clearing in time (`ProbClearInTime`):

$$P_{\text{ok}}(\kappa) = \sum_{w \in W} \pi\,(w) \sum_{b} P\,(b \mid \kappa, w)\, \mathbb{1}[\,M(\kappa, w, b) \geq \beta\,].$$

Admissibility.

$$\mathcal{A} = \{\,\kappa: P_{\text{ok}}(\kappa) \geq \alpha \text{ and } C(\kappa) \leq B\,\}.$$

A.3 The Evacuation Model

A discrete-time system with step $\Delta = 1$ min, $t \in \{0,1, \ldots, H\}$. State at time t:

- $q_v^o(t) \geq 0$ — vehicles of origin o waiting at intersection v (a cohort)

- a set of in-transit cohorts $(t_{\text{arr}}, v, o, m)$: m vehicles of origin o arriving at v at time t_{arr}
- $\text{blocked}(e,t) \in \{0,1\}$ for $e \in E$
- crew state (position, current street, remaining work) — §A.3.3
- $\text{informed}(o,t) \in \{0,1\}$ per origin o — §A.4

Initial condition: $q_v^v(0) = n_v(\theta_w)$, all other $q = 0$; nothing in transit; blocked $\equiv 0$; $\text{informed}(o,0) = \mathbb{1}[\, b_w = A \text{ and } t_{\text{fall}}(w) = 0\,]$.

Routing in force for cohort o at v:

$$r_t(v,o) = \begin{cases} r(v) & \text{if informed}(o,t) = 0, \\ \tilde{r}_t(v) & \text{otherwise,} \end{cases}$$

where $\tilde{r}_t$ is the shortest-time routing to X on the graph $(V, \{e \in E : \text{blocked}(e,t) = 0\})$, and $\tilde{r}_t(v) = \perp$ if no path exists.

Each step $t \to t+1$ executes, in order:

A.3.1 Line fall (item 3 of the main text). If $t = t_{\text{fall}}(w)$: $\text{blocked}(e,t) \leftarrow 1$ for all $e \in E_s$, $s \in F$; the crew is dispatched (§A.3.3).

A.3.2 Flow (items 1–2 of the main text).

(a) Arrivals: every in-transit cohort with $t_{\text{arr}} \le t$ is added to q_v^o.

(b) Exits: for $x \in X$, with $Q = \sum_o q_x^o$, move $\min(Q, c_x/60)$ vehicles out, proportionally over origins; record exits by origin and time.

(c) Streets: for each non-exit v and each street $e = \{v, w\}$ with $w = r_t(v,o)$ for some cohort o and $\text{blocked}(e,t) = 0$, let

$$Q_e = \sum_{o:\, r_t(v,o) = w} q_v^o ;$$

move $\min(Q_e, c_e/60)$ vehicles, proportionally over those cohorts, into transit with $t_{\text{arr}} = t + \ell_e$.

A.3.3 Repair-crew motion and repair (item 4 of the main text). The crew occupies an intersection. When it begins crossing street $e = \{u, w\}$ from u toward w at time t_0, the crossing completes at

$$t_0 + \ell_e + \frac{Q_e^{\leftarrow}(t_0)}{\rho_e},$$

where $Q_e^{\leftarrow}(t_0)$ is the number of vehicles on e at t_0 flowing toward u (queued at w with $r_{t_0}(w,\cdot) = u$, plus in transit toward u). On reaching an endpoint of a street in E_s for a fallen span s, repair begins; it completes after $\rho(s)$ minutes, whereupon $\text{blocked}(e,\cdot) \leftarrow 0$ for $e \in E_s$ and the crew selects the nearest (fewest streets, over unblocked streets) remaining fallen span and proceeds. Hence

$$T_{\text{rep}}^{\text{eff}}(s) = T_{\text{arr}}(s) + \rho(s),$$

with $T_{\text{arr}}(s)$ (`CrewArrivalMinutes`) a function of the network state between X and s, not of s alone (`EffectiveRepairMinutes`).

A.3.4 Resident behaviour update (item 5): see §A.4.

A.3.5 Termination (item 6).

$$T_{\text{clear}} = \min\left\{ t \le H : \sum_{v,o} q_v^o\,(t) = 0 \text{ and nothing in transit} \right\}, \quad \text{or } H \text{ if no such } t.$$

Determinism: the map $(\kappa, w, b) \mapsto T_{\text{clear}}$ is a function; all randomness is in b and w and is enumerated in §A.5.

A.4 Resident Behaviour

$b_w \in \{U, L, A\}$ determines informed(o, t):

b_w	Rule	Main-text name
U	informed$(o,t) = 0$ for all o, t	`Uninformed` (default)
L	informed$(o, t+1) = 1$ if informed$(o,t) = 1$, or if $q_v^o(t) > 0$ at some v with blocked$(\{v, r(v)\}, t) = 1$; else 0	`LocalKnowledge`
A	informed$(o,t) = 1$ for all o and all $t \ge t_{\text{fall}}(w)$	`Alerted`

Decision rule for the choice of b_w: $b_w = U$ unless an alert or traffic-control plan is on record, in which case L or A may be used and the plan is published with the result.

A.5 Probability Computation

For fixed (κ, w) the set of blockage states is enumerated as a tree over the controlling spans $s_1, \ldots, s_k$ ($k = |S_c|$), with the per-span branch probabilities of (A.2):

```
Enumerate(κ, w):
    L ← ∅ ;  U ← 0                              -- leaves, unresolved mass
    Expand(i = 1, F = ∅, ρ = ∅, p = 1)
    return (L, U)

Expand(i, F, ρ, p):
    if p < ε:    U ← U + p ; return          -- ProbabilityFloor
    if i > k:    L ← L ∪ {(F, ρ, p)} ; return
    Expand(i+1, F, ρ, p · (1 − p_{s_i}(κ(s_i), w)))                    --
branch A
    for r in R_{s_i}(κ(s_i)):                                           --
branches B, C, …
```

```
Expand(i+1, F ∪ {s_i}, ρ ∪ {s_i ↦ r},
       p · p_{s_i}(κ(s_i), w) · f_{s_i}(r | κ(s_i)))
```

Reductions applied before enumeration: spans with $K_s = \emptyset$ are omitted; spans with identical E_s and identical f_s are merged into one group with fall probability $1 - \prod(1 - p_s)$.

Merging by outcome: for each leaf, run the model of §A.3 to obtain T_{clear}; the clearance-time pmf for (κ, w) is

$$g_{\kappa,w}(t) = \sum_{(F,\rho,p)\in L\,:\,T_{\text{clear}}(\kappa,w,(F,\rho))=t} p.$$

Safe leaves (`SafeLeaves`) and the per-scenario probability:

$$L_{\text{safe}}(\kappa, w) = \{\, (F,\rho,p) \in L : T_{\text{fire}}(w) - T_{\text{clear}}(\kappa, w, (F,\rho)) \geq \beta \,\},$$

$$P_{\text{ok}}(\kappa \mid w) = \sum_{(F,\rho,p)\in L_{\text{safe}}(\kappa,w)} p,$$

$$P_{\text{ok}}(\kappa) = \sum_{w\in W} \pi(w)\, P_{\text{ok}}(\kappa \mid w).$$

Lower-bound property: the true probability equals $P_{\text{ok}}(\kappa)$ plus the safe part of the unresolved mass, so (A.12) understates the probability of clearing in time by at most $\sum_w \pi(w)\, U(\kappa, w)$; the error is in the conservative direction.

A.6 Decision Algorithm

Let κ_{UG} be the configuration with $\kappa(s) = \text{ug}$ for all $s \in S_c$ (`FullUnderground`) and κ_{H} the one with $\kappa(s) = \text{hard}$ for all $s \in S_c$ (`AllHardened`). Downgrading one step: ug → hard → bare.

Step 1 — Feasibility. If $P_{\text{ok}}(\kappa_{\text{UG}}) < \alpha$: report the scenarios w and blockage states b with $M(\kappa_{\text{UG}}, w, b) < \beta$, N_{trap} for each, and the zones $\bigcup_s K_s$ involved; halt (infeasible by line mitigation alone).

Step 2 — Budget. If $C(\kappa_{\text{UG}}) > B$: proceed to Step 3; if $\mathcal{A} = \emptyset$ at its end, report

$$\text{BudgetShortfall} = \min\{\, C(\kappa) : P_{\text{ok}}(\kappa) \geq \alpha \,\} - B.$$

Step 3 — Least-cost admissible configuration. If $k \leq 12$ (exhaustive):

$$\kappa^* = \operatorname*{argmin}_{\kappa\in\mathcal{A}} C(\kappa), \qquad \text{ties broken by } \max P_{\text{ok}}(\kappa).$$

Else (monotone descent): set $\kappa \leftarrow \kappa_{\text{UG}}$ and repeat:

- for each $s \in S_c$ with $\kappa(s) \neq$ bare, let κ_s be κ with s downgraded one step;
- if $\kappa_s \in \mathcal{A}$, set $\text{Value}(s) = \frac{C(\kappa) - C(\kappa_s)}{\max(P_{\text{ok}}(\kappa) - P_{\text{ok}}(\kappa_s),\, \delta)}$;

- if no $\kappa_s \in \mathcal{A}$: stop; else $\kappa \leftarrow \kappa_{s^*}$ with $s^* = \text{argmax}_s \text{Value}(s)$;

until stop; $\kappa^* \leftarrow \kappa$. ($P_{\text{ok}}$ is non-increasing under downgrades, so every visited κ is admissible and the loop terminates in at most $2k$ iterations.)

Step 4 — Record statement. For $\kappa \in \{\kappa^*, \kappa_{\text{UG}}, \kappa_{\text{H}}\}$ compute $P_{\text{ok}}(\kappa)$;

$$P_{\text{neg}}(\kappa) = \sum_w \pi(w) \sum_b P(b \mid \kappa, w)\, \mathbb{1}[M(\kappa, w, b) < 0]$$

(`ProbNegativeMargin`); the 5th-percentile margin $M_{0.05}(\kappa) = \sup\{m : \Pr[M \le m] \le 0.05\}$; the failing (w, b) with N_{trap}; the expected trapped-vehicle-minutes $\mathbb{E}\left[\int_{T_{\text{fire}}}^{H} (\text{vehicles inside at } t)\, dt\right]$; and $C(\kappa)$.

Step 5 — Cost-benefit among admissible configurations.

$$\text{Cand} = \{\kappa \in \mathcal{A} : C(\kappa) \le (1+\tau)\, C(\kappa^*)\}, \qquad \kappa_{\text{final}} = \arg\max_{\kappa \in \text{Cand}} \text{BCR}(\kappa),$$

where BCR is the utility's ratio of present-value ignition and outage benefits to present-value cost. Admissibility (A.5) is a hard constraint; BCR acts only within Cand.

Step 6 — Publication. The tuple (inputs of §A.1; (L, U) and $g_{\kappa,w}$ for $\kappa \in \{\kappa^*, \kappa_{\text{UG}}, \kappa_{\text{H}}\}$ and all w; the Step 3 trace; the Step 4 statement; κ_{final}) is published. Reproducibility requirement: the map from inputs to κ_{final} is deterministic.

A.7 Notional Use Case

$X = \{x_{\text{main}}, x_{\text{narrow}}\}$, $c_{x_{\text{main}}} = 900$ veh/h, $c_{x_{\text{narrow}}} = 450$ veh/h; $\sum_v n_v = 4{,}220$; $S_c = \{s_1 = \text{MainApproach},\ s_2 = \text{NarrowLeg},\ s_3 = \text{UpperLane}\}$ with $|K_{s_3}| = 60$ households (≈ 110 vehicles); $T_{\text{fire}} = 240$, $\beta = 20$, $\alpha = 0.99$; T_{clear} with $F = \emptyset$: 206 min.

κ	$(\kappa(s_1), \kappa(s_2), \kappa(s_3))$	$C(\kappa)$	$P_{\text{ok}}(\kappa)$	Status
κ_{UG}	(ug, ug, ug)	3.0	1.000	admissible
κ^*	(ug, ug, hard)	2.3	1.000	argmin$C$ over $\mathcal{A}$
—	(ug, hard, hard)	1.7	0.970	$P_{\text{ok}} < \alpha$
κ_{H}	(hard, hard, hard)	1.0	0.880	$P_{\text{ok}} < \alpha$

Statement (Step 4): $P_{\text{ok}}(\kappa^*) = 1.00$ at $C(\kappa^*) = 0.77\, C(\kappa_{\text{UG}})$; $P_{\text{ok}}(\kappa_{\text{H}}) = 0.88$.

Sensitivity: $T_{\text{fire}} = 360 \Rightarrow P_{\text{ok}}(\kappa_{\text{H}}) \approx 0.995 \ge \alpha$, so $\kappa^* = \kappa_{\text{H}}$ (hardening admissible and cheapest). Dendritic network, single exit, $T_{\text{fire}} = 150 \Rightarrow \mathcal{A} = \{\kappa_{\text{UG}}\}$.

A.8 Correspondence of names and symbols

Main text	Symbol	Defined in

Main text	Symbol	Defined in
`RequiredClearProbability`	α	A.1.1
`SafetyBufferMinutes`	β	A.1.1
`Budget`	B	A.1.1
`PlanningHorizonMinutes`	H	A.1.1
`ProbabilityFloor`	ε	A.1.1
`CostTolerance`	τ	A.1.1
`RoadNetwork`	$G = (V, E)$	A.1.2
`LengthMinutes`, `Lanes`, `StreetCapacity`, `CrewPassRate`	$\ell_e, \lambda_e, c_e, \rho_e$	A.1.2
`Exits`, `ExitCapacity`	X, c_x	A.1.2
`VehiclesAt[v]`	$n_v(\theta)$	A.1.2
`NormalRoute[v]`	$r(v)$	A.1.2
`Spans`, `HouseholdsStrandedIfDown`, controlling spans	S, K_s, S_c	A.1.2
`Option`	$o \in O = \{\text{bare}, \text{hard}, \text{ug}\}$	A.1.3
`Cost[span, option]`	$\text{cost}(s, o)$	A.1.3
`ProbSpanFalls[span, option, scenario]`	$p_s(o, w)$	A.1.3
`RepairWorkMinutes[span, option]`	$f_s(r \mid o),\ R_s(o)$	A.1.3
`ScenarioSet`, `ScenarioWeight`	$W, \pi(w)$	A.1.4
`FireArrivalMinutes`, `LineFallMinutes`	$T_{\text{fire}}(w), t_{\text{fall}}(w)$	A.1.4
`ResidentBehaviour`	$b_w \in \{U, L, A\}$	A.1.4, A.4
`Configuration`, `ConfigurationCost`	$\kappa, C(\kappa)$	A.2, (A.1)
`BlockageState` and its probability	$b = (F, \rho),\ P(b \mid \kappa, w)$	A.2, (A.2)
`ClearanceMinutes`	$T_{\text{clear}}(\kappa, w, b)$	A.2, A.3
`EgressMargin`	M	(A.3)
`ResidentsTrapped`	N_{trap}	A.2
`ProbClearInTime`	$P_{\text{ok}}(\kappa)$	(A.4), (A.12)
`Admissible`	$\kappa \in \mathcal{A}$	(A.5)
`CrewArrivalMinutes`, `EffectiveRepairMinutes`	$T_{\text{arr}}(s),\ T_{\text{rep}}^{\text{eff}}(s)$	(A.7), (A.8)
`SafeLeaves`	$L_{\text{safe}}(\kappa, w)$	(A.10)
`UnresolvedProbability`	U	A.5
`FullUnderground`, `AllHardened`, `Selected`, `Final`	$\kappa_{\text{UG}}, \kappa_{\text{H}}, \kappa^*, \kappa_{\text{final}}$	A.6
`BudgetShortfall`	(A.13)	A.6
`Candidates`	Cand	(A.15)

Main text	Symbol	Defined in
`ProbNegativeMargin`, worst 1-in-20 margin	$P_{\text{neg}}(\kappa),\ M_{0.05}(\kappa)$	A.6 Step 4

Appendix B — Simulation Models and Experimental Frames

RTSync Corp. — October 2026

This appendix describes the software with which the results in this document were produced, the experiments run on it, their verification, and their findings. It is written for the modelling team and for any party wishing to reproduce or extend the results. Section B.1 explains the three technologies used; B.2 the models; B.3 the experimental frames; B.4 verification; B.5 the illustrative networks and parameters; B.6 the results of each frame; B.7 how to run the algorithm on a named community; B.8 limitations.

B.1 Technologies

Discrete-event simulation and DEVS. The evacuation model advances from event to event — a cohort of vehicles leaves a street, a span falls, a repair completes — rather than in fixed increments of wall-clock time. It is written in DEVS (Discrete Event System Specification), a mathematical formalism in which a model is a tuple of input and output ports, states, a time-advance function and transition functions (Zeigler, Muzy & Kofman, *Theory of Modeling and Simulation*, 3rd ed., 2018). Writing the model in DEVS makes its interface explicit, lets models be composed by coupling ports, and lets the experiment that drives the model be separated from the model itself. Note that the model is entirely framed in code neutral DEVS formalism and can be implemented in any DEVS compliant platform. The code used for this initial case study targets the MS4 Me DEVS environment (MS4 Systems, ms4systems.com).

ParaDEVS (paratemporal DEVS). The uncertainties in the problem are discrete: a span falls or it does not, and if it falls the repair takes one of a few durations. Rather than sampling these at random (Monte Carlo), a ParaDEVS computation follows every branch, carrying each branch's probability as the product along its path, and merges branches that lead to the same outcome. The result is the exact probability distribution of the outcome for the model, with no sampling error. This is what makes a tail probability such as "clears in

time in 99 of 100 events" computable to the precision the decision needs, and what makes an exhaustive search over mitigation configurations affordable (B.6.4).

EFSUT (Experimental Frame – System Under Test). Each regulatory question is answered by a declared experimental frame: a *Generator* that produces the inputs the model is fed, a *Transducer* that observes its outputs and computes the measures, and control conditions saying when the experiment stops. Frames are organised in a partial order by derivability — a coarser frame's results can be obtained from a finer frame's by selecting ports, aggregating, or coarsening — and each frame is matched to the simplest model that can answer it (Zeigler, *EFSUT*, Digital Engineering and Digital Twin 4, 2026). RTSync's ef-builder workflow generates the frames as compilable DEVS code with built-in verification checks, so the record of each decision is a set of declared experiments rather than an analyst's narrative.

B.2 The models

All models are in package `Models.java`, project `ResidentExitTime`.

B.2.1 `EgressNetwork` — the road network builder

Constructs a community network from a specification string and assigns deterministic edge identifiers shared by the model and the frames.

Topology	Specification	Structure
Grid	`GRID,rows,cols,demand=d`	Lattice; exits at the two bottom corners; many alternative paths
Tree (dendritic)	`TREE,depth,branch,demand=d`	Rooted tree; root is the main exit; one extra edge joins the last leaf to a secondary exit, so exactly one bypass exists

Default parameters: street travel time 2 min, street capacity 600 veh/h, main exit 900 veh/h, secondary exit 450 veh/h, demand `d` vehicles per non-exit node. Methods provide shortest-time routing to the nearest exit over the unblocked graph, the set of controlling edges, and the exit-approach edges.

B.2.2 `ResidentExitTimeModel` — the evacuation network (SUT)

An atomic DEVS model. Input ports `topology`, `scenario`, `evac_order`; output ports `exit_times` (per origin: first and last exit minute and count), `clearance_report` (community clearance, residents still inside at fire arrival, scenario echo), `queue_status` (every 10 min). States `IDLE → READY → DRAINING → CLEARED → READY`.

Flow is computed minute by minute on cohorts (vehicles grouped by origin): each street passes at most its capacity, each exit at most its capacity, vehicles move one street per minute at most and mix proportionally across cohorts at each queue. A scenario names the spans that fall, the fall time, and the repair duration of each.

Routing is per cohort. A cohort is either *uninformed* (follows its normal route and waits at a block) or *informed* (follows the shortest open route). Which cohorts are informed, and when, is set by the resident-behaviour mode of the scenario (main text §4): `DIRECTED` — none; `LOCAL` — a cohort becomes informed when it stands at a street it can see is blocked; `UNDIRECTED` — all cohorts at the moment of the fall.

The model is deterministic: given topology and scenario, it always returns the same clearance time.

B.2.3 `CrewAccessExitTimeModel` — the repair crew as a mobile resource

Extends the model above by overriding its block-update step only. One crew leaves the main exit when the lines fall. To reach a span it crosses streets one at a time against the outbound flow; crossing a street takes its free-flow time plus the number of vehicles queued on it divided by the crew pass rate (30 vehicles per minute at a slowdown factor of 1; divided by the slowdown factor otherwise). On arrival it repairs for the span's repair duration, then selects the nearest still-blocked span reachable over open streets. The scenario gains a `slowdown` key; the clearance report gains `crew_travel`, `crew_work`, `last_repair` and `crew_pass` fields. With slowdown 1 and empty streets it reproduces the parent model.

B.2.4 `ParaDEVSBlockageTree` — the enumeration

Given the candidate spans, their fall probabilities, the repair-duration distribution and a probability floor, enumerates the blockage tree depth-first: at each span, one branch for "does not fall" and one per repair duration for "falls"; branches below the floor are pruned and their mass accumulated as unresolved. Each surviving leaf becomes one scenario carrying its path probability.

B.3 The experimental frames

Two frames were developed, each as the standard set of five files (Generator, Transducer, flat coupled frame, Generator-Transducer sub-model, hierarchical coupled frame).

B.3.1 `EF_ExitTimeStudy` — exit times and their exact distribution

The maximal frame of the study. Its Generator has four base rows — {grid, tree} × {uninformed, informed} — and for each expands a ParaDEVS tree over the candidate spans (grid: the four exit-approach streets; tree: the six streets at depth one and two), issuing one scenario per leaf with its probability: 99 + 99 + 314 + 314 = 826 scenarios. Its Transducer merges leaves by outcome into the exact clearance-time distribution per base row and computes the probability of clearing in time, the probability of a negative margin, expected residents trapped, and per-node expected first and last exit times. It answers four stakeholder questions:

Question	Derived frame

Question	Derived frame
Each resident's exit time under normal conditions, by location	`EF_DirectedGrid`, `EF_DirectedTree` (selection)
How blocked streets change exit times under the directed first approximation, grid vs tree	same
What bypasses and crew access (the second approximation) add	`EF_BypassComparison` (selection)
The exact distribution and the probability of not clearing in time	`EF_ExitTimePMF` (coarsening)

B.3.2 `EF_CrewAccessSweep` — when crew access dominates

A deterministic sweep against `CrewAccessExitTimeModel`: {uninformed, local-knowledge} × {grid, tree} × {baseline, 1, 2, 3 repair sites} × crew slowdown {1, 2, 4, 8, 12, 16, 24, 32} = 100 scenarios. Repair sites are interior streets so the crew must pass queued traffic; with two sites the second lies behind the first on the same route. The Transducer records clearance, crew travel, crew work, last repair time and vehicles passed for each cell, and flags a cell as *access-dominated* when the crew's travel exceeds its work time *and* the clearance is longer than at slowdown 1 for the same number of sites — that is, when the crew's slowness, not the block itself or the exit capacity, lengthened the evacuation.

B.4 Verification

The ef-builder workflow applies two verification passes.

Conceptual verification constructs the maximal frame (the union of all ports and scenarios across candidate frames) and checks that every frame's ports exist on the model, that the combined scenarios reach every declared state, and that every frame is derivable from the maximal one. Both frames pass.

Developmental verification checks the generated code: port declarations against the maximal frame, every accessor called by a frame against the model, that every output port is emitted in some state, coupling counts (7 for the flat frame, 7 for the sub-model, 6 for the hierarchical frame), and that scenarios drive the declared state sequence. Both frames pass.

Execution. Each component has a standalone test, and each coupled frame a full run with at least eight pass/fail checks. Results:

Test	Outcome
Model, Generator, Transducer standalone (both frames)	all pass
`EF_ExitTimeStudy`, flat and hierarchical: 826 runs, 536,442 simulated minutes, 1.5 s wall clock	10 of 10 checks pass, including: observed plus unresolved probability sums to 1; no-block baseline identical under both behaviours; grid bypasses do not worsen the negative-margin probability; tree

Test	Outcome
	blockages lengthen clearance; first exits grow with depth
`EF_CrewAccessSweep`, flat and hierarchical: 100 runs	8 of 8 checks pass, including: crew travel and clearance increase with slowdown; crew work and last repair increase with sites; no blocked run is faster than baseline at slowdown 1; an access-dominated region exists
Re-run of `EF_ExitTimeStudy` after the per-cohort routing refactor	results identical to the first run

Model-level checks: with no blockage the model reproduces hand-computed clearance (demand divided by exit rate, plus travel) and, with one blockage, the closed-form hold-versus-reroute thresholds of the earlier two-exit study, to within ±2 minutes.

B.5 Illustrative networks and parameters

Item	`EF_ExitTimeStudy`	`EF_CrewAccessSweep`
Grid	4 × 4, 16 nodes, 24 streets, exits at nodes 12 (900 veh/h) and 15 (450 veh/h)	same
Tree	depth 3, branching 2, 16 nodes, 15 streets, root exit 900 veh/h, secondary exit 450 veh/h via one bypass edge	same
Demand	100 vehicles per node	200 vehicles per node
Line fall	20 min after the order	10 min
Fall probability per candidate span	0.15 (hardened conductor)	— (sites fixed)
Repair durations	30 (0.4), 60 (0.4), 120 (0.2) min	30 min per site
Fire arrival	120 min (grid), 240 min (tree)	—
Probability floor	10^{-4}	—
Candidate spans	grid: 4 exit approaches; tree: 6 streets at depth 1–2	grid: streets 4–8, 5–9, 6–10; tree: 1–3, then 3–7 behind it, then 2–6

These are illustrative networks chosen to exhibit the two extremes of street structure; their parameters are deck conventions, not survey values.

B.6 Results

B.6.1 Exit times and the effect of topology (`EF_ExitTimeStudy`)

Base row	No-block clearance	Clearance min / mode / max	P(clear in time), lower bound	P(negative margin)	E[residents trapped]
Grid, uninformed, fire at 120	96	96 / 96 / 196	0.827	**0.167**	31.1
Grid, informed	96	96 / 96 / 150	0.975	**0.019**	4.3
Tree, uninformed, fire at 240	89	89 / 89 / 203	0.974	0.000	0.0
Tree, informed	89	73 / 89 / 275	0.973	0.001	0.1

Unresolved (pruned) mass 0.6 % (grid) and 2.6 % (tree); the clear-in-time probabilities are lower bounds by those amounts.

- On the grid, information cuts the negative-margin probability from 0.167 to 0.019, because the other exit approaches absorb a blocked one.
- On the tree, information changes almost nothing (one bypass exists), and sequential repair-crew access lengthens the worst case from 203 to 275 minutes.
- Every tree street is critical for the residents above it: a block anywhere strands them until repair, although their last-exit times are otherwise set by the root bottleneck, not by depth.
- Nearest-exit routing leaves the tree's secondary exit idle in the baseline; a block that forces traffic onto it can shorten clearance (73 < 89). The assignment of residents to exits is itself a decision variable.

B.6.2 Crew access and cascades (`EF_CrewAccessSweep`, residents uninformed)

Tree, baseline 176 min (bold = access-dominated):

Sites	s = 1	2	4	8	12	16	24	32
1	176	176	176	176	**212**	**251**	**328**	**406**
2 (behind the first)	176	176	**183**	**275**	**366**	**459**	**> 600**	**> 600**
3	176	**194**	**237**	**329**	**420**	**513**	**> 600**	**> 600**
Crew travel, 1 site	12	21	41	79	118	157	234	311
Crew travel, 2 sites	28	51	96	188	280	372	556	740

Grid, baseline 189 min: 189 up to slowdown 4, then **194 / 232 / 271 / 348 / 426** at slowdowns 8–32, for every number of sites.

- Below a threshold the crew's slowness costs nothing: the trapped group is released early enough to be absorbed by the exit bottleneck. Above it, clearance grows one minute per minute of crew travel. For one site the threshold is slowdown 12 on the tree (a pass rate of 2.5 veh/min, about 24 s per car) and 8 on the grid.
- A second site behind the first drops the threshold to slowdown 4 (8 s per car): the crew passes the first queue, repairs, then passes a second queue that has kept growing (690 vehicles passed instead of 290). At slowdown 24 and beyond the community is not clear within ten hours.
- The grid offers no protection if residents wait at the block.

B.6.3 The value of information (same sweep, local-knowledge behaviour)

With cohorts that turn around on seeing a block: the tree rows for one and two sites are unchanged (no detour exists); the three-site row improves by 21 minutes (the cohort at the third block turns to the idle secondary exit); the grid shows no crew effect at any slowdown (one site: 189 throughout; two or three sites: 284, a capacity penalty independent of the crew). Fully alerted behaviour did no better than local knowledge on either network.

B.6.4 Exact enumeration versus Monte Carlo

For a probability p estimated by N random trials, the standard error is $\sqrt{(p(1-p)/N)}$; resolving p to ±10 % needs $N \approx (1-p)/(p \cdot 0.01)$. For the decision-grade quantity — a negative-margin probability near the 1 % requirement — that is about 10,000 trials per configuration, against 99–314 enumerated leaves per base row here; for the tree's 10^{-3} tail, about 100,000; to reproduce the full distribution to the 10^{-4} floor, about 10^{6}. At the measured 1.3 ms per run, enumeration took 1.5 s for the whole study. Monte Carlo cost grows as 1/p, exactly where the requirement places the outcomes of interest; enumeration cost is fixed by the number of candidate spans.

B.7 Running the algorithm on a named community

13. Build the road network from county centreline GIS: intersections, streets with length, lanes, capacity (from lane count and local counts), exits with capacity. Assign a crew pass rate per street from lane count and parking (B.2.3).
14. Assign vehicles per intersection from parcel data and vehicles per household, for at least two times of day.
15. Obtain the span inventory with street crossings from the utility's circuit-segment data; compute each span's stranded households and the controlling set.
16. Estimate per-span fall probabilities by wind class from tree-strike density (LiDAR), conductor type and pole age, and the repair-duration distribution from the utility's wind-event repair logs.
17. Run fire-spread simulation (DEVS-FIRE, Technosylva Wildfire Analyst, or FARSITE/FlamMap) over the declared ignition points and wind days to obtain fire-arrival times; correct fuel maps for local vegetation.

18. Set the scenario set and weights from wind climatology; set resident behaviour to uninformed unless an alert plan is on record.
19. Generate the frames for the community with ef-builder: `EF_Feasibility`, `EF_Budget`, `EF_Efficiency`, `EF_Risk`, `EF_Sensitivity`, each a declared Generator–Transducer pair over the same model, and execute in MS4 Me.
20. Publish the inputs, the leaf tables, the search trace and the statement (main text §6, Step 6).

B.8 Limitations

- Vehicles are tracked as cohorts by origin with proportional mixing at queues; there is no per-vehicle identity, and "each resident's exit time" is resolved to the street.
- Streets have one capacity regardless of direction; contraflow use of inbound lanes is not represented except through the crew pass rate.
- The crew is a single resource starting at the main exit; multiple crews, staging points, and crew safety limits (crews will not enter active fire) are extensions.
- The probability floor leaves a small unresolved mass; probabilities are reported as lower bounds, and the floor can be lowered at modest cost.
- The networks in B.5 are illustrative; every parameter is to be replaced by surveyed values before a decision is made for a named community.
- Execution has been verified against a stand-in for the MS4 runtime; confirmation in MS4 Me is required.

B.9 Files

File	Content
`ResidentExitTime/java/EgressNetwork.java`	Network builder (B.2.1)
`ResidentExitTime/java/ResidentExitTimeModel.java`	Evacuation model (B.2.2)
`ResidentExitTime/java/CrewAccessExitTimeModel.java`	Crew extension (B.2.3)
`ResidentExitTime/java/ParaDEVSBlockageTree.java`	Enumeration (B.2.4)
`ResidentExitTime/java/ResidentExitTime_ExitTimeStudy_*.java`	Frame B.3.1 (five files)
`ResidentExitTime/java/ResidentExitTime_CrewAccessSweep_*.java`	Frame B.3.2 (five files)
`ResidentExitTime/docs/ResidentExitTime_EF_Analysis.md`	Conceptual-phase report with addenda
`ResidentExitTime/docs/ResidentExitTime_EFSUT_Table.md`	Question →

File	Content
	frame → model table
`ResidentExitTime/docs/ResidentExitTime_Verification_Report.md`	Verification record
`ResidentExitTime/docs/CrewAccessSweep_run.txt`	Full sweep transcript

References and Sources

- NFPA Fire Investigation Report, Oakland/Berkeley Hills Fire
- U.S. Fire Administration Technical Report, The East Bay Hills Fire
- State OES Multi-Agency Review of the East Bay Hills Fire
- CFIW Oakland/Berkeley Hills report
- Oakland North, "Twenty years after the Oakland hills fire"
- Montclair resident's account, "The '91 Firestorm"
- Wikipedia, Oakland firestorm of 1991
- KQED on the 2025 Berkeley evacuation study
- City of Berkeley Evacuation StoryMaps (KLD study results)

- U.S. Department of Defense. (2018). *Digital Engineering Strategy*. Office of the Deputy Assistant Secretary of Defense for Systems Engineering, Washington, DC. Retrieved from https://apps.dtic.mil/sti/pdfs/AD1068564.pdf
- Zeigler, B. P. (2024). Discrete Event Systems Theory for Fast Stochastic Simulation via Tree Expansion. *Systems*, 12(3), 80.
- Zeigler, B. P. (2026a). Experimental Frame–System Under Test (EFSUT): A Principled Foundation for Model Choice and Lifecycle Management in Digital Twins. Digital Engineering and Digital Twin, 4, 1–26.
- Zeigler, B. P. (2026b). Specifying Simulation Commitments Early in System Design: Introduction to the EFSUT Methodology. *Systems*, 14(6), 691.
- Zeigler, B.P.; Woon, S.W., Koertje,C.and Zanni, C., The Utility of Homomorphism Concepts in Simulation: Building Families of Models from Base-Lumped Model Pairs. Simulation J. 2024, in process.
- .Zeigler, B.P. Constructing and evaluating multi-resolution model pairs: An attrition modeling example. J. Déf. Model. Simul.: Appl. Methodol. Technol. 2017, 14, 427–437. https://doi.org/10.1177/1548512917717085.

- Zeigler, B. DEVS-Based Building Blocks and Architectural Patterns for Intelligent Hybrid Cyberphysical System Design. Information 2021, 12, 531. https://doi.org/10.3390/info12120531.

- Zeigler, B.P.; Muzy, A.; Kofman, E. Theory of Modeling and Simulation: Discrete Event Iterative System Computational Foundations; Academic Press: New York, NY, USA, 2018

- Zeigler, B.P.; Seo, C.; Kim, D. DEVS Modeling and Simulation Methodology with MS4 Me Software. In Proceedings of the DEVS 13: Proceedings of the Symposium on Theory of Modeling & Simulation—DEVS Integrative M&S Symposium, San Diego, CA, USA, 7–10 April 2013.