Politecnico di Milano
Computer Science Engineering

Formal Methods for Concurrent and Real-Time Systems

Formal Analysis of Search-and-Rescue Scenarios

Homework Report

Students
Fiano Michael - 10676595
Lamperti Federico - 10680961
Salonico Davide - 10774487

Academic Year 2023-2024

Contents

1 Introduction

The project involves creating a model to describe an autonomous rescue service, based on three main entities:

- Civilian
- Drone
- First responder

Drones support first responders (professional rescuers) by detecting people in need of help while flying over the scene. In this report, we will present the modeling choices we made and provide a brief description of the results we achieved through our tests and various scenarios.

2 Design

2.1 Layout

We modeled the map as a grid of arbitrary size, where each entity is represented by a different number (e.g. a cell occupied by fire is represented by the number 2). Civilians and First responders can move in any cell unless it is already occupied by another entity or by fire. Civilians aim to reach an exit as quickly as possible, while first responders move on the map seeking civilians in danger (adjacent to a cell occupied by fire) to rescue them. In contrast, drones are not represented on the map since they fly over it without any issues. Collisions between drones are avoided by providing deterministic trajectories. More details will be provided in the subsequent sections.

Only for the stochastic version of the model, we modeled the following stochastic features simply using the UPPAAL built-in branch tool to create probabilistic edges:

- Drones' vision sensor fault
- Probability of civilians not acknowledging drones' instructions

The values of these probabilities are provided in completely tunable global arrays named respectively `drone_faults[]` and `civilian_ignores[]`and the values are assigned to each entity in the scenario when initialized.

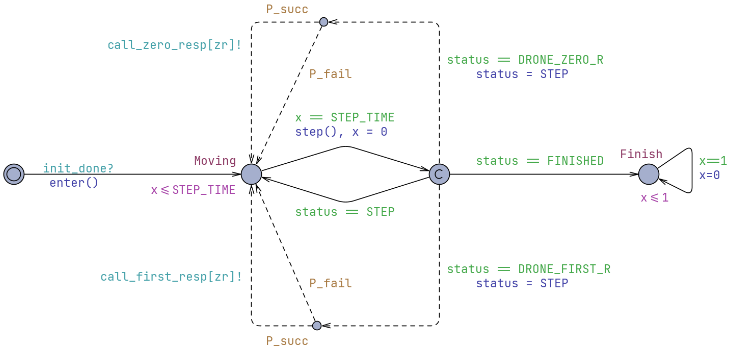


Figure 1: TA for the Drone in the stochastic version

2.2 Assumptions

The following assumptions were made at the design stage and must be satisfied for the model to be simulated correctly and meaningfully:

- Drones' trajectories are intended to scan the entire map as efficiently as possible.
- None of the entities can enter a cell occupied by fire.
- An exit cell cannot also be occupied by fire.
- Entities can also move by performing diagonal steps.
- When a civilian is safe or dies, he is removed instantly from the map.
- When either a zero or a first responder is busy because they have been called, they are considered busy for a specific amount of time and do not move in the map.
- Drones have full knowledge of first responders' status and they can communicate (modeled as reading/writing global variables).

3 UPPAAL model & Design Choices

3.1 Global declaration

3.1.1 System parameters

`GRID_LENGTH` and `GRID_HEIGHT`	represent the dimensions of the grid
`N_EXITS`	the number of exits on the boundary of the map
`N_FIRES`	the number of fires on the map
`N_DRONES`	number of drones in the scenario
`N_FIRST_RESPONDERS`	number of first responders in the scenario
`T_ZR`	the time that a Zero Responder needs to help someone
`T_FR`	the time that a First Responder needs to help someone
`T_V`	the time before a civilian is supposed to die
`MAX_TRAJ`	the maximum trajectory length

3.1.2 Variables

- `grid[GRID_LENGTH][GRID_HEIGHT]`: a 2D matrix representing the entire scenario, where each cell contains a value indicating what is occupying that specific cell. This is what defines scenario's geometry, but values contained in the matrix are compliant with system parameters.
- `drone_trajectories[MAX_TRAJ][N_DRONES]`: stores the predefined trajectories that drones must follow.
- `drone_ranges[N_DRONES]`: stores the drones' visibility ranges.
- `exits[N_EXITS]`, `fires[N_FIRES]`, `pos_FRs[N_FIRST_RESPONDERS]` and `pos_civilians[N_CIVILIANS]`: store the position of the different entities in the map. It's a redundant representation but always coherent with the grid. These arrays are used by utility functions and they make computations lighter.
- `saved` and `casualties`: store the number of civilians who were saved or died. Used to evaluate properties.
- `caller_id`: represents the ID of the civilian instructed to call a first responder.
- `pos_to_save`: store the position of the civilian to be saved to a generic responder (drones write and zero/first responders read)

The use of global variables for `caller_id` and `pos_to_save` is due to the impossibility of passing variables over UPPAAL channels. This mechanism is an abstraction of the physical channel from drones to human used in the real rescue scenario.

3.2 Channels

For synchronization purposes, we defined the following channels:

- `init_done`: broadcast signal used only for initialization to signal all templates to start.
- `helping`: signal sent by either a zero responder or a first responder to a civilian in danger.
- `saved_by_ZR`: a broadcast signal that indicates that the time needed for a zero responder to save the civilian has elapsed.
- `saved_by_FR`: a broadcast signal that indicates that the time needed for a first responder to save the civilian has elapsed.
- `FR_help_req`: an urgent signal sent by a civilian to a first responder in case is instructed to do so, by setting the channel to urgent, we prioritize the message received.
- `call_zero_resp`: the drone signals a civilian to become a zero responder, the channel is urgent to avoid multiple detections by different drones.
- `call_first_resp`: the drone signals a civilian to call a first responder, the channel is urgent to avoid multiple detections by different drones.
- `civ_dead`: a broadcast signal informing that the civilian has become a casualty, indicating no need for further assistance.

Not all the channels strictly require to be broadcast for the model to function correctly, however since it's required to have only broadcast channels to perform statistical model checking queries we didn't change between the two models for uniformity.

3.3 Templates

3.3.1 Initializer

This is a basic timed automaton with a single transition that occurs instantly because the initial state is committed. The `init_all()` function creates the map and sets up global variables. The `init_done` channel signals all other timed automatons in the system to start their execution.

3.3.2 Civilian

This automaton models the behavior of civilians. Civilians enter a committed state that checks the status of the civilian and initializes his position in the grid. If the starting position is already near an exit, the civilian is considered safe and removed from the map. If instead, he is near a cell occupied by fire, the civilian enters the Danger state, where he can't move waiting for assistance (helping message from another civilian or first responder) and consequently becoming safe or waiting for its clock to arrive at `T_V` and become a casualty, thus leaving the map sending a `civ_dead` signal. If none of these conditions apply, the civilian is free to move and receive messages from drones:

- If a safe or danger condition is satisfied, the civilian changes his status accordingly.
- If a `call_zero_resp` message is received from a drone, he is considered busy (changing the global status on the map to avoid multiple calls) and sends a helping message to the civilian in need. If the rescue is successful, a message `saved_by_ZR` is sent to the civilian in need and both civilians are considered safe, otherwise if the civilian in need dies before the end of the recovery, he returns to the moving state.
- If a `call_first_resp` message is received from a drone, he is considered busy (changing the global status on the map to avoid multiple calls) and sends an `FR_help_req` to the nearest First responder available. If the rescue is successful, a message `saved_by_FR` is sent to the civilian in need and both civilians are considered safe, otherwise if the civilian in need dies before the end of the recovery, he returns to the moving state.

3.3.3 Drone

This automaton models the behavior of drones. After the initialization, the drone enters the moving state and performs a step following its trajectory. Then it scans the environment within its visibility range and updates its status if both a civilian in danger and another civilian not already occupied are found, using a policy that will be illustrated in the next section. The drone can change its status to `DRONE_ZERO_R` and subsequently dispatch a `call_zero_resp` message to the identified free civilian, or to `DRONE_FIRST_R` thereby sending a `call_first_resp` message. Otherwise, it continues to move. Finally, when every civilian is either safe or dead, it goes to a sink state that represents the end of the scenario.

3.3.4 First Responder

This automaton models the behavior of first responders. After receiving the `init_done` message, a committed state checks if a civilian in danger is nearby, if so the responder becomes busy and proceeds to the helping state, sending a helping message to that civilian. Otherwise, he goes to the moving state, where he will move within the grid until he's near a civilian in danger, or receives an `FR_help_req` message, prompting him to move to the helping state. In the helping state when the time elapses a `saved_by_FR` message is sent to the assisted survivor and the responder returns to the Moving state, becoming available again. If in the meantime a `civ_dead` message is received, the responder becomes available and returns to the Moving state. Finally, when every civilian on the map is either safe or dead, the responder enters a sink state that represents the end of the scenario.

3.4 Policies

3.4.1 Civilians' moving policy

For the civilians, we decided to set 2 different policies:

- `RANDOM`: every civilian moves randomly on the map, which is realistic as panic can overtake reason in a dangerous situation or visibility could be compromised.
- `CLOSER_EXIT`: each civilian searches for the nearest exit and heads directly to it, this policy is also realistic as survival instinct guides people in unsafe situations.

3.4.2 Drones' moving policy

As previously mentioned, we provided drones with a deterministic trajectory for every scenario. Realistically, drones will be programmed to scan the environment as efficiently as possible, exploiting the known characteristics of the environment, such as dimensions and exit locations. In our model trajectories are completely customizable and they can have different shapes and lengths for each drone in the scene. This is one the features of our model that assures scalability, a leading principle during the whole development process

3.4.3 Drones' decision policy

Drones must choose whether to instruct a zero responder or to call for a first responder. As stated in the assumptions, it is very likely that First responders are somehow connected with drones and they can interact, displaying each other's current status. We modeled the choice of which policy to apply by first checking if at least a first responder is available. If not the most efficient way to save people is to opt for a zero responder. If at least one first responder is not busy, we implemented four different policies to follow:

- `RANDOMLY`: the drone decides randomly who to call.
- `ALWAYS_ZR`: dummy policy that consists of calling always a zero responder.
- `ALWAYS_FR`: dummy policy that consists of calling always a first responder when at least one is free.
- `MIN_TIME`: the drone predicts the total time needed in both cases to save the civilian in danger and makes the most efficient choice, which is the one with a shorter predicted time.

3.4.4 First responders' moving policy

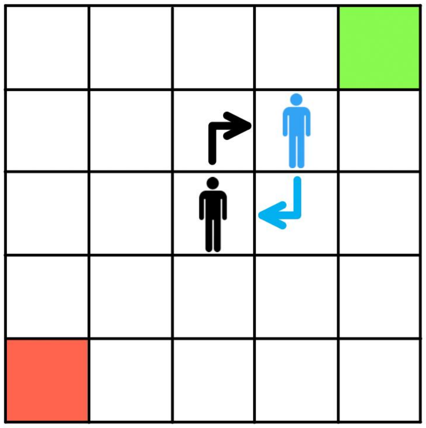
- `RANDOM`: In a realistic scenario, first responders should move throughout the map attempting to help as many people as possible. Thus, a random policy can be appropriate.
- `FIRES`: First responders are trained personnel, hence they try to maximise the number of rescues. Since fire is what make people perish in our problem, first responders go to the closest fire in the scene. If they're not instructed to save anyone and they're close to a fire, it means that probably there is no one to save in their nearby and consequently, they direct close to another fire in the scene.

> The use of a random policy, implemented by mean of select on edges in UPPAAL, leads to a state number explosion during the *Verify* phase. For this reason we created a different folder in the deliveries to show the implementation of a model with non-deterministic policies but the required properties for these mentioned models are not supposed to be verified for time constraints (even if theoretically possible). To check that the properties hold please refer to the version without RANDOM policies (folder phase1).

3.5 Conflicts

Two entities may try to move to the same cell at the same time. To solve this problem we structured the entities to first check if a cell is not occupied and if so move immediately to that location updating the global grid.

This conflict is implicitly resolved by UPPAAL's atomicity of transitions. Since transitions in timed automata are executed atomically (even if they occur at the same instant), only one entity will move to the target cell at any given time, thereby preventing the conflict. If an entity cannot move to any adjacent cell, it does not perform the step. With such simple but efficient modeling conflicts are solved, but deadlocks are introduced when using certain movement policies. If a first responder and a civilian try to move in opposite directions each of them will see the next cell on his path occupied and they would stay still forever. Hence, if the first responder detects an obstacle on his way he can perform a different move toward his direction even if this implies increasing (of one step) the remaining path. This different behaviour is enough to solve the deadlock since this situation can't occur among two civilians and ,in addition, the choice of enhancing the "capabilities" of the first responder only is motivated by the fact that they have more expertise on the field and they're capable of taking critical choices. Scenario 2 shows how this case is correctly solved. We provide the code of the function dealing with first responder's alternative step in case of the `FIRES` movement policy.



(a) Simple example of a potential deadlock

🟩	Exit
🟥	Fire
🧍	Civilian
🧍	First Responder
✠	Drone

(b) Legend for graphical representation of a scenario

```c
pos_t getAlternativeNext(pos_t next){
    pos_t alt1 = OUT_OF_MAP;
    pos_t neighbors[3][3];
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if (pos.r + i - 1>= 0 && pos.r + i - 1< GRID_HEIGHT && pos.c + j - 1>= 0 && pos.c + j - 1< GRID_LENGTH){
                neighbors[i][j].r = pos.r + i - 1;
                neighbors[i][j].c = pos.c + j - 1;
            }
            else neighbors[i][j] = OUT_OF_MAP;
        }
    }
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(neighbors[i][j] != OUT_OF_MAP && isCellFree(neighbors[i][j]) && dist(target_fire, neighbors[i][j]) <= (1 + dist(target_fire, next))){
                alt1 = neighbors[i][j];
            }
        }
    }
    return alt1;
}
```

4 Properties

From requirements we had to check if the following two properties hold:

- it is possible for a percentage N% of all civilians to reach a safe state within time T_scs;
- a percentage N% of all civilians is always guaranteed to reach a safe state within time T_scs.

where N% (sometimes written as N_perc) and T_scs are tunable parameters that vary based on the scenario. For phase 2 (Statistical Model Checking) we were required to express the probability of the aforementioned properties holding within specifc time bounds. The properties, written in TCTL, become:

P1: $\forall\Diamond\left(\text{total\_time} \leq T_{\text{scs}} \wedge \left(\frac{\text{saved}\times 100}{N_{\text{CIVILIANS}}} \geq N_{\text{perc}}\right)\right)$

P2: $\forall\Box\left(\text{total\_time} = T_{\text{scs}} \rightarrow \left(\frac{\text{saved}\times 100}{N_{\text{CIVILIANS}}} \geq N_{\text{perc}}\right)\right)$

in the following section we evaluated results obtained analyzing different scenarios trying to explain how parameters can affect the evolution of a scenario (hence changing, even significantly) N% and T_scs. Moreover we verified additional properties to ensure the correctness of our model:

- System in deadlock only when a scenario is finished (every civilian is either safe or dead)
- Number of civilians in danger always more or equal to number of busy FRs plus number of busy ZRs
- Number of civilians in helping state always equal to FR busy + ZR busy
- The scenario will end

These three last properties are always verified (also for the stochastic version) as they're not dependent on the particular configuration of the scene. In the next session we discuss the result we obtained in different scenarios. Notice that deadlock here is only a way to model the end of the episode and it's not caused by an error in the model.

5 Analysis and results

We performed a detailed analysis for different configurations, trying to highlight the main characteristics of each scenario and which are the parameters that play a crucial role for the properties to be verified. In order to speed up the scenario generation process we wrote a python script to automatically generate a valid configuration. The usage is very simple and intuitive and it's not intended as a perfect scenarios generator but as a way to save time and effort automating the major part of the process. In the script all the main parameters present in the Global Declaration section of the model are provided as input and a grid with fires, exit, civilians and first responders is created. In addition, drones and their trajectories are generated with a basic pattern. In order to create a valid configuration that could resemble a real scenario, fires are placed using a combination of cluster and random sampling. Exits are randomly sampled from cells along the borders of the grid. Civilians and first responders are placed randomly on the grid. The following scenarios (except from the example already provided in class) are generated using the script and eventually changing some details of the configuration by hand. In the following scenarios, whenever not stated differently, we used `MIN_TIME` decision policy for drones, `CLOSER_EXIT` for civilians' movement policy and `FIRES` for first responders' movement policy.

5.1 Scenario 1: Example

This configuration just resembles the same scenario provided in the requirements document. Drones' have different "circular" trajectories and the direction (clockwise or anticlockwise) are fixed for each drone. Using `T_FR`=5, `T_ZR`=3 and `T_V`=8 the properties show that the 80% of the civilians in the scene are always guaranteed to be saved within 11 time units.

5.2 Scenario 2: DeadlockSolved

This scenario shows how a potential deadlock between a civilian and a first responder is correctly managed and the end of the episode is reachable in a finite number of time steps. The following figure [2a] shows an instance of this situation: the civilian is directed to the exit while the first responder is directed on the direction of the fire, they both see next cell on their path as occupied. As previously mentioned the first responder is able to adapt changing his path. In addition, in this example is possible to notice how the order of the rules is respected by the model: if a person is both adjacent to an exit and a fire he must be considered as saved and not in danger.
In this scenario only 50% of the civilian can be saved in 7 time units using same parameters as above. This difference w.r.t. the first scenario is justified by a lower number of drones in the scene and by non optimal drones' trajectories. In fact, the only two drones present scan the grid along two horizontal lines going back and forth for the entire duration of the episode[Fig 3].

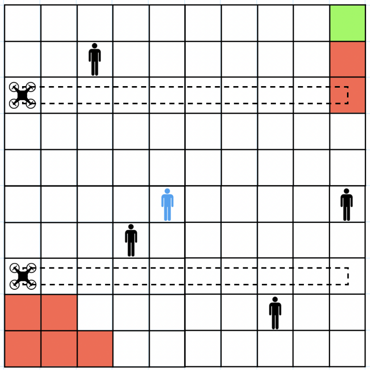


Figure 3: Scenario 2: DeadlockSolved

5.3 Scenario 3: AllFRBusy

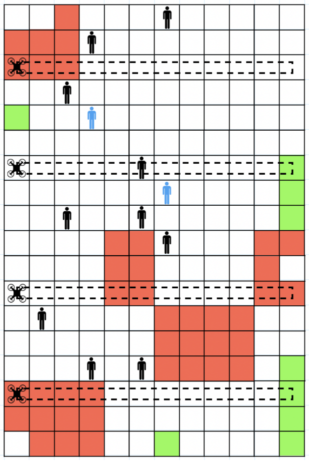
This scenario shows a situation in which the drone is instructed to always call a first responder if it detects a person in danger and a civilian in its range. However if all the first responders are busy and another person is in need the drone is still able to instruct a civilian to help the person in difficulty (hence becoming a zero responder). In this scenario the drones adopt the `ALWAYS_FR` policy in order to force the situation in which they are all busy. Also the grid layout is not square, showing that every rectangular configuration is possible. The scenario has been tested with the following parameters: `T_FR`=11, `T_ZR`=7 and `T_V`=16.

As a result only 30% of the civilians in the scene are guaranteed to be saved in 23 time units. This great difference highlights how time parameters (and a non optimal decision policy) play a crucial role for the evolution of the episode.

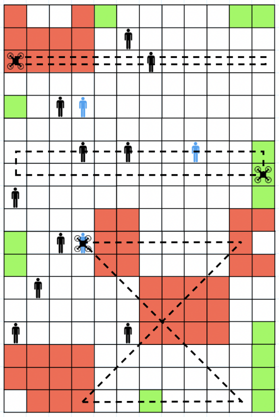
5.4 Scenario 4: DifferentDrones

In this configuration we still use a rectangular layout but the key difference with respect to the previous ones is the presence of different parameters and trajectories for the drone. The first one goes back and forth horizontally, the second follows a rectangular trajectory and the third one goes in a personalized path. The visibility ranges for the drones are different too, proving the capacity of the model to manage completely independent drones in the same scenario. In addition, the decision policy of the drones is `ALWAYS_ZR`. The following figure is a graphic representation of the described situation.

In this situation, 50% of the civilians are always saved within 6 time units. Again this non ideal result can be justified by the non optimality both in the drones' trajectories and in their decision policy.



(a) Scenario 3: AllFRBusy



(b) Scenario 4: DifferentDrones

5.5 Random Policy

As mentioned before an extensive model checking for non-deterministic policies is not feasible. Hence, we provide some simulation results along with some considerations for the first three scenarios. Only in the first case we provided the detailed result for each simulation, for the following scenarios a summary is provided for the sake of readability.

Scenario 1: Example

Despite the randomness, the number of survivors in the scenario tends to remain similar across all simulations. This is due both to the initial configuration and to the movement of the drones, which effectively hover over the only fire present in the scenario. Regarding the time, the situation is much more variable. This is due to the non-optimal paths taken by the civilians, who often "spin around" without heading towards the exit. It is worth noting the significantly higher average time compared to the use of other policies. Also noteworthy is how the best time approaches the results obtained with optimal policies.

simulation	saved	casualties	total_time
1	6	3	32
2	8	1	39
3	8	1	43
4	8	1	13
5	6	3	28
6	8	1	35
7	8	1	25
8	8	1	23
9	7	2	28
10	7	2	45
MEAN	7.4	1.6	31.1
VARIANCE	0.711	0.711	95.878
BEST	8	1	13
WORST	6	3	45

Scenario 2: DeadlockSolved

In this case, the number of civilians who survive is much lower compared to the non-random counterpart. This is due to the fact that the only exit in the scenario is located in a corner of the map, further surrounded by fires, making it very unlikely for a civilian to approach without putting themselves in danger. Additionally, despite the drones flying over areas very close to the fires, the number of casualties remains very high due to the low number of civilians involved and the likelihood that these civilians are spread out, thus increasing the rescue time. As for the very high times, this is certainly due to the fact that the map is very empty, giving civilians few opportunities to change their state.

simulation	saved	casualties	total_time
MEAN	0.8	3.2	124.6
VARIANCE	1.289	1.289	5360.933
BEST	3	1	72
WORST	0	4	324

Scenario 3: AllFRBusy

Based on the scenario we created, two civilians start off in danger but are instantly rescued by two first responders (FR). The simulation results are very similar because many civilians start in close proximity to the fires and are therefore in a state of danger. The fires are also widely present on the map, increasing the likelihood of civilians entering a state of danger. Since it is very easy for civilians to enter a state of danger, the average time is relatively low.

simulation	saved	casualties	total_time
MEAN	5	5	48
VARIANCE	0.667	0.667	587.778
BEST	6	4	17
WORST	3	7	92

5.6 Statistical Model Checking

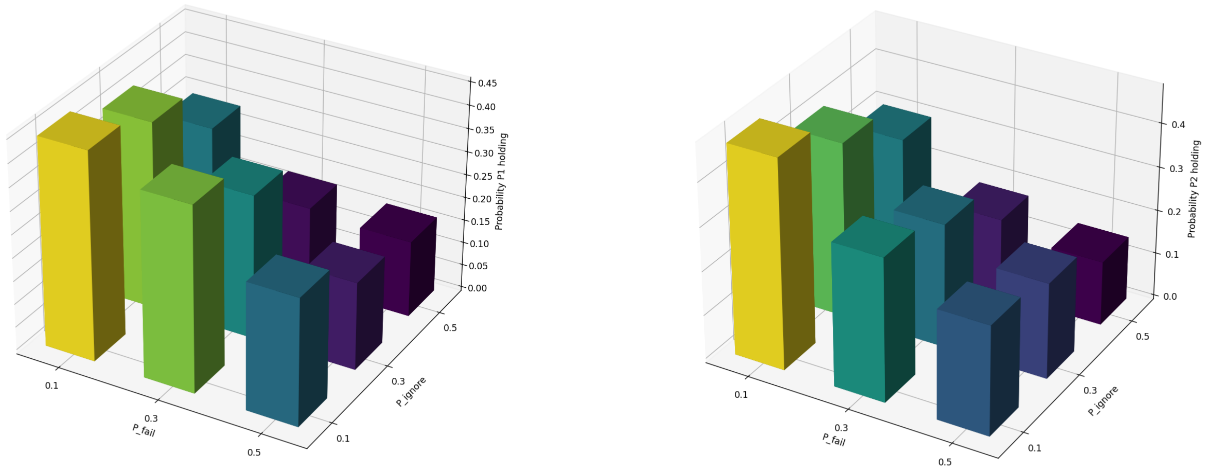
For the analysis of the stochastic version of the model we chose to show the same scenarios as above (same layout and policies), each with different probabilities for `P_ignore` and `P_fault`. Whenever we show results with a certain `P_ignore` or `P_fault` we refer to the mean value of the probability distribution respectively over civilians and drones. Differently from the model with deterministic transitions we added a loop transition in the sink state of Drone to explicitly model the passing of time without deadlocks to let SMC[1] queries work correctly. This leads to infinite time episodes but it's a problem since SMC is bounded by definition. We wrote queries to check the probabilities of mandatory properties as follows:

P1 SMC: `Pr[<=1000] (<> ((total_time <= T_scs) and ((saved * 100 / N_CIVILIANS) >= N_perc)))`
P2 SMC: `Pr[<=1000] ([] (total_time == T_scs) imply ((saved * 100 / N_CIVILIANS) >= N_perc))`

We maintained also additional queries but, as expected, since they don't depend on SMC features, they always produced a probability over 95% (maximum). The results obtained for the mandatory properties are commented below. The confidence level is set to 95% to estimate confidence interval. The estimated variation is always less than 5% and it's not reported not to aggravate readability of results.

Scenario 1: Example

In this scenario, failure rates affect probability result in a significant way. The more a drone is faulty or a person is prone not to listen, the less probable will be to guarantee a rescue for the majority of the civilians. Setting `T_scs` = 12 and `N_perc` = 100 the following results were achieved.



(a) P1

(b) P2

Figure 5: 3D-Histograms representing probabilities of P1 and P2 holding for different configurations

Other scenarios

For the other scenarios, we noticed that probabilities of failures/not listening don't play an important role in determining the number of saved civilians. This is because either many of the entities exit the scene almost immediately or the first responders autonomously find people in danger, hence signaling mechanism is not crucial. We show results for scenario 2 and 3 [Tab 1].

6 Conclusions

The model has been designed trying to resemble as close as possible a real case scenario, letting the user customize many of the parameters in a scene. In addition, we tried to make the model extensible to further improvements, whether they are the introduction of new entities in a scenario or the addition of new policies.

P_fail	P_ignore	P1	P2
0.1	0.1	0.068	0.046
0.3	0.3	0.106	0.093
0.5	0.5	0.093	0.036

(a) SMC results for Scenario 2 with T_scs=30 and N_perc=75

P_fail	P_ignore	P1	P2
0.1	0.1	0.805	0.827
0.3	0.5	0.836	0.813
0.5	0.5	0.830	0.827
0.3	0.3	0.818	0.820
0.3	0.5	0.772	0.866
0.5	0.5	0.820	0.784
0.5	0.5	0.788	0.816
0.3	0.5	0.773	0.755
0.5	0.5	0.716	0.376

(b) SMC results for Scenario 3 with T_scs=17 and N_perc=50

Table 1: Results for other scenarios showing quasi-uniform probabilities

For instance, the decision policy is decoupled from the step action, hence making the model simple and scalable.

References

[1] Alexandre David, Kim Larsen, Axel Legay, Marius Mikučionis, and Danny Poulsen. Uppaal smc tutorial. *International Journal on Software Tools for Technology Transfer*, 17, 01 2015.