POLITECNICO
MILANO 1863

Formal Methods For Concurrent
and Real-Time Systems

FORMAL ANALYSIS OF SEARCH-AND-RESCUE SCENARIOS with UPPAAL

Homework Project

Authors
Valentino Guerrini
Antonio Marusic
Patrick Poggi

Instructors
Prof. Pierluigi San Pietro
Dr. Livia Lestingi

A.Y. 2023-2024

Abstract

This report explores a real-time simulation of a search and rescue scenario where first responders help civilians escape a fire and reach the exits safely. The rescue operation is aided by drones that patrol the area. The focus of the model is to understand the survival possibilities given different fire and exit layouts, as well as varying numbers of civilians and first responders.

1 Introduction

The idea behind this topic is to understand how autonomous systems, such as drones, can be utilized to aid the evacuation process during natural disasters (e.g., the fire scenario we covered). This scenario is highly safety-critical, making it worthwhile to spend time modeling it for better understanding. UPPAAL allows for this modeling through the formalism of a Network of Timed Automata (NTA), which implements the asynchronous interaction between different Timed Automata.

In section 2 we give a brief introduction to each one of the actors of our simulation, then the in depth description of each one of them, their interaction and the list assumptions of the model is covered in section 3. The analysis of the model is then described in section 5.

2 High level model description

Our Uppal model includes four templates:

- **Initialize:** This template is used only at the beginning of the simulation and aims at initializing the grid by placing all elements involved in the model. First it sets fires, drones and exits, then it positions civilians and first-responders, one by one. Furthermore, it sets the paths for drones and First Responders.
- **Civilian:** Civilians move inside the grid and can be harmed by the fire, in which case they will need assistance. If they aren't brought to a safe state within a predefined time they are considered a casualty. Moreover, if they stand next an exit they are considered safe and removed from the grid. Civilians can be instructed by drones to either act as zero responders and help a victim, or to contact a first repsonder to do so.
- **Drone:** Each drone patrols the grid according to a specific moving policy looking for civilians. Upon finding a civilian in need of assistance and a civilian not in need, the drone can either instruct the sound civilian to act as a zero responder and help the one in need or to do so by contacting a first responder.
- **First Responder:** The purpose of First Responders is to travel around the grid in search for civilians in need of assistance (i.e., that have got next to a fire) and help them. Alternatively, first responders can be contacted by civilians that have been instructed by drones to go and help a victim.

These templates simulate the search and rescue of civilians in a grid with fires and exits. Exits are always at the grid's borders, while fires occupy contiguous cells. Figure 7 shows a valid grid layout. Actors move only up-down or left-right, one tile per time unit. Civilians and first responders cannot share a cell, but drones can share cells with people since they are not ground-constrained. responder.

3 Component description

Here, we describe each template implemented and mention the data structures used.

3.1 Data structures

3.1.1 Grid

The grid has been modeled with three matrices:

```
int grid[GRID_WIDTH][GRID_HEIGHT]
meta int path[GRID_WIDTH][GRID_HEIGHT]
meta int fr_path[GRID_WIDTH][GRID_HEIGHT]
```

The grid matrix stores the position of fires, exits, and civilians. Since those entities cannot share the same position, a single matrix is enough to represent them. The two path matrixes stores the drones and first responder's predefined paths, represented with an integer value associated with each possible direction (i.e., up, down, left, right). The keyword **meta** is used because in UPPAAL, this ensures that the variable is not counted in the state space. This is particularly useful for the path matrix, as the predefined paths followed by the drones remain constant and do not affect state exploration.

3.1.2 Data types

```
typedef struct {
    int[-1,GRID_HEIGHT-1] row; // range specifies the valid range for row
    int[-1,GRID_WIDTH-1] col;  // range specifies the valid range for col
} position_t;
```

This is used to indicate a position on the grid.

```
typedef struct {
    int[1,4] d; // direction of the next move
    position_t p; // position
} direction_t;
```

This is used to create the path that the drone follows.

Initialization and Configuration Parameters :

Parameter	Type	Description
GRID_WIDTH	const int	Width of the grid
GRID_HEIGHT	const int	Height of the grid
FIRE_NUM	const int	Number of fires on the grid
EXIT_NUM	const int	Number of exits on the grid
CIVILIAN_NUMBER	const int	Number of civilians
FIRST_RESPONDER_NUMBER	const int	Number of first responders
DRONE_NUM	const int	Number of drones
PATH_LEN	const int	Size of the array with the drones' paths
FR_PATH_LEN	const int	Size of the array with the first responders path
STEP_TIME	const int	Duration of a time unit for agents to move
Nv	int[DRONE_NUM]	Range of sight of the drones
Tzr	int[CIVILIAN_NUMBER]	Time necessary to rescue a needy civilian when the civilian becomes a zero responder
Tv	int[CIVILIAN_NUMBER]	Time until a needy civilian dies
Tfrs	int[FIRST_RESPONDER_NUMBER]	Time necessary for the first responder to rescue a needy civilian
drones_positions	position_t[DRONE_NUM]	Stores the position (coordinates) of each drone
drones_path	direction_t[PATH_LEN]	Marks the paths of the drones.
firstResponder_pos	position_t[FIRST_RESPONDER_NUMBER]	Initial positions of the first responders
first_responders_path	direction_t[FR_PATH_LEN]	Marks the paths of the first responders.
civilians_positions	position_t[CIVILIAN_NUMBER]	Initial positions of the civilians

Table 1: Simulation Parameters

Query Parameter	Type	Description
rescued_civilians	int	Number of civilians that successfully exit the grid
casualties	int	Number of civilians that die on the grid

Table 2: Parameters Used in Queries

Stochastic Parameter	Type	Description
Pfail	int	Probability that the sensor fails to communicate with a civilian
Plisten	int	Probability that a civilian listens to the indications of a drone
civ_not_listened	int	Number of times some civilians have not listened the drone
drone_faults	int	number of faults of the drone

Table 3: Stochastic Parameters

Assumptions

1. The paths of the drones **don't overlap**.
2. The parameters are **correctly initialized**, namely the coordinates of the various elements are positive or null and inside the dimension of the grid specified with the GRID_WIDTH and GRID_HEIGHT and the NUM/NUMBER constants are coherent with the number of entities declared
3. The paths of the first responders are **correctly initialized**.
4. Any actor who is **rescuing** another one will **not get harmed** itself neither while traveling towards the victim nor while actually performing the rescue.

Implementation notes

1. We use channels to synchronize the involved actors to simulate the communication between the various entities. Then, they exchange that information using **globally accessible arrays** that are indexed with the **ID of the entity**. For example, if a civilian of id x has to access information that he is expecting, he will access the position id of the array where he will find the information.
2. We didn't explicitly model the movement of the zero responders and first responders towards the needy civilians; we just **simulated** it by making them wait in status for the time that would have been necessary for the movement.
3. When the drone instructs the civilians and first responders those are set to a special state that marks them as **busy** so that two drones can't contact the same person.
4. When the drone instructs a civilian to contact a first responder, as said, the civilian, the first responder and the needy civilian will be set to the special status "**RESCUING**". When the rescuing process finishes the needy civilian and the zero responder are brought to a safe state by making them disappear from the map, while the first responder takes the victim's place.

5. The fact that the movement cannot happen simultaneously, thus risking conflicts for the position in a cell, is granted by Uppal's atomicity of the transaction. The same property grants that two drones cannot contact the same people.

3.2 Initialize

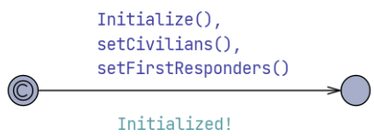

Figure 1: Initializer automata.

This template is used to initialize all data structures to their initial value. The primary data structure is the **grid** as it is the environment actors move in. This is initialized by placing **fires**, **exits**, **drones** and their **path(s)**, **civilians** and **first-responders** as described in the configurable parameters inside the global declarations. The second step is civilian placement. When the initialization process has terminated, a broadcast synchronization message is sent. This message activates the first transition for all the other templates and initiates the simulation.

3.3 Civilian

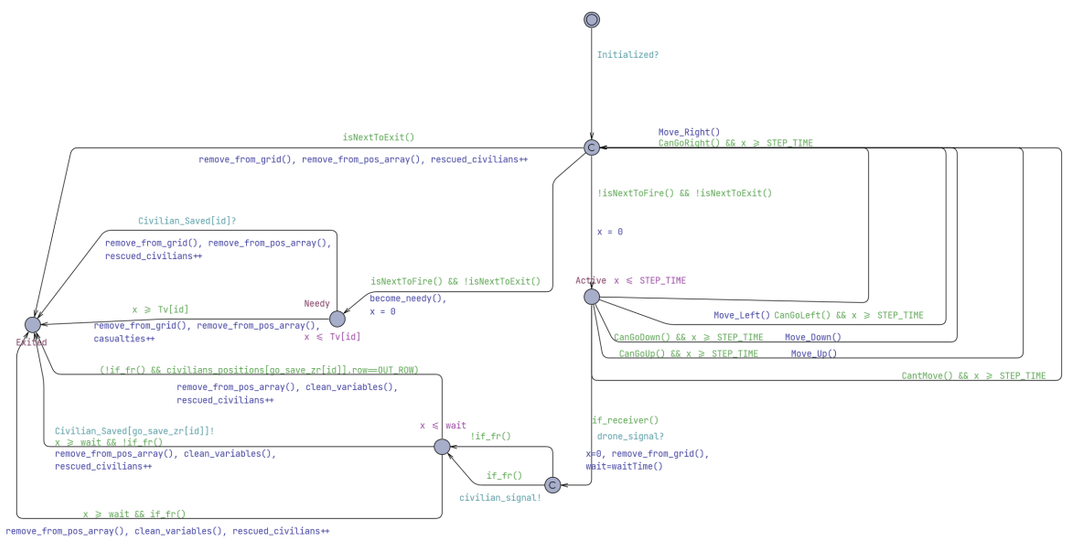

Figure 2: Civilian automata.

Figure 2 shows the timed automata that model civilians' behavior. The civilian, after the initialization of the grid, goes into a committed state from which he can perform different transitions and move into one of the following states:

1. **Exited**: if the civilian is next to an exit, he will exit the map. We model this behavior by setting the position of the civilian outside the map.

2. **Needy**: If the civilian is close to a fire but not to an exit he will transition to Needy and hold its position for **Tv** time units. If instead the civilian is close both to a fire and to an exit, the aforementioned case will be valid as per project specification. From this state the civilian can either be saved by a rescuing agent or become a casualty if that doesn't happen within **Tv** time units. The civilian is notified it's been saved through the channel '`Civilian_Saved`'.

3. **Active**: if the civilian is neither next to an exit nor to a fire, he will hold its position for `STEP_TIME = 1` time units and then take step in the way described below.

If he is in the active state, he will remain there for 1 time unit, and one of the following things can happen:

1. **The civilian moves**: The civilian can move in one of the four possible directions (up, down, left, right) for a single cell if the cell exists (i.e., is within the bounds of the map) and it is not occupied by another ground element (i.e., either a person or an exit or a fire) and if the new position is in a

better position than the one he is into, meaning that the distance to the closest exit is less than the current one. We decided to implement a movement policy that is more intelligent than just moving non-deterministically because a person in danger won't move away from the exit and wander on the map. However, this movement policy may cause the civilian to move toward a fire if the fire is between him and the nearest exit.

2. **The civilian doesn't move**: If the civilian is unable to move due to the presence of obstacles or because, as mentioned, any available move would take it further away from an exit, it will stand still and wait for this condition to change.

3. **The civilian is contacted by a drone to become a zero responder**: using the 'go_save_zr' data structure along with the 'drone_signal' broadcast channel, the drone tells the civilian to become a zero responder and also indicates the id of the civilian in need of assistance. In this case, the civilian will be set in a busy sate on the map in order to simulate its unavailability with respect to drones. By performing this transition the civilian will move into a state in which it will wait a time equal to the *Manhattan* distance between its position and the needy's one, plus the time needed to pay assistance. During this process we assume the civlian acting as a Zero Responder cannot become needy itself by encountering fire. After this time, which models both the traversal of the grid for the rescue and the time taken by the rescue itself, the civilian will be considered rescued and exit the grid. If the victim has not become a casualty, the civilian will inform it that it's been saved and can exit the grid.

4. **The civilian is instructed by a drone to call a first responder**: By using the go_call array along with the 'drone_signal' broadcast channel, the drone can inform the civilian to reach a first responder and instruct him to help the needy civilian. The civilian will, in this case, wait a time given by the sum of the *Manhattan* distances between him and the First Responder and between the FR and the needy civilian (indicated by the drone by means of the 'go_save_fr' array), plus the time for the rescue. After such time has elapsed the civilian will be considered safe and thus exit the grid. Like in the previous case we assume the civlian won't become needy while traveling towards the first responder it has to contact. In this case, the notification about the victim being saved is forwarded by the FR instead of by the civlian.

   Whenever a civilian is involved in a rescuing mission it can happen that the victim becomes a casualty before being able to be saved, in which case the victim will not be counted as a rescued civilian and the rescuing one will stop paying assistance and safely exit the grid.

3.4 FirstResponder

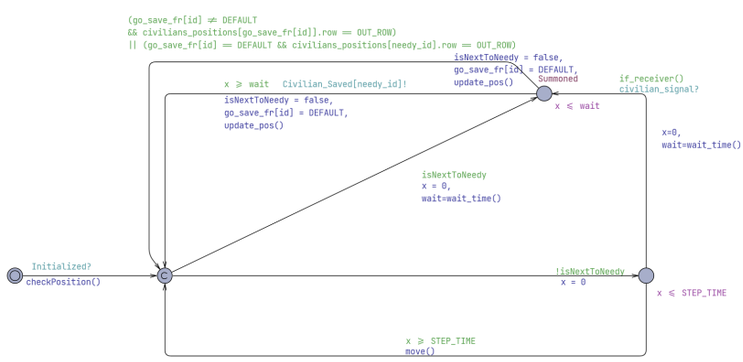

Figure 3: First responder automata.

Each first responder waits in the initial state until the grid is initialized. Then, it moves into a committed state from which it is able to detect any needy civilian in its surroundings and act consequently, as described below.

1. **civilian found**: In this case, the first responder goes into the *Summoned* state, waiting for the time necessary for the rescuing operations (i.e., `Tfr`. If the civilian is successfully rescued, a message is sent on the '`Civilian_Saved[civilian_id]`' channel (where 'civilian_id' is the id of the saved civilian) and then the FR will update its position to that of the rescued civilians, from which it will restart its normal operation. Otherwise, if the civilian dies before the first responder can conclude the rescue, the FR will get back to its starting position and check again its surroundings, continuing its default activity.

2. **civilian not found**: If a needy civilian is not found, the first responder enters a state in which it will stay for `STEP_TIME` time units. When such time elapses, it can move to a different location. In the meanwhile, it can be contacted by civlians that have been instructed by the drones. Civilians perform such contacting procedure through the '`civilian_signal`' broadcast channel ad the '`go_call`' array. From this state, as soon as `STEP_TIME` time units have elapsed, the FR will move according to its moving policy.

FR's moving policy: We have designed FRs' **moving policy** in order to simulate them patrolling the area near to the fire by circling around it, as they are experienced entities which can assess the higher risk of the fire surroundings. This has been implemented through a static parameter in the global declaration that indicates the FRs' path, which they follow. If a FR is **unable to follow the predefined path**, it will momentarily move in the only direction that is neither along the path in either of the two ways or **momentarily exits the path**. Whenever a FR successfully rescues a victim it will **take its position**, which may be **away from the predefined path**, it will **move towards** the path to start **following it again**.

3.5 Drone

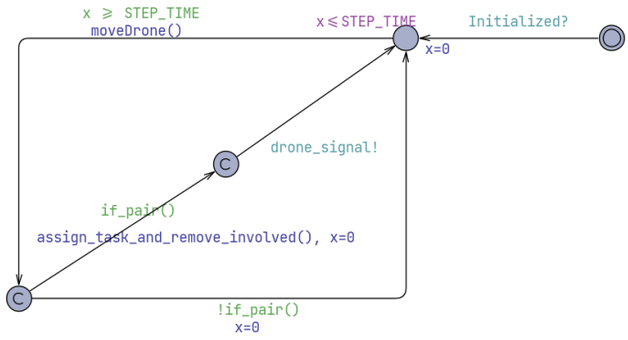

Figure 4: Drone automata.

Drones patrol the grid according to a **predefined configurable path**. Whenever they find a pair of civilians not in need and a civilian in need, they can either instruct the one not in need to help the other by himself or call a first responder. This decision is based on whether there is a first responder **between the civilian and the victim** in the drone's visibility range. If that's the case, the drone will choose to have the civilian engage the first responder to better model a **realistic** behavior, whereas if that is not the case, the first option is chosen. Suppose more then one first responders can help needy civilians in the drone's visibility range. In that case, it chooses the one for which the sum of the distance between the civilian and the FR and the distance between the FR and the victim is the **smallest one**. In the case that no FR is present, the only option the drone can choose is to instruct the civilian to act as a zero-responder. In both cases, the drone's job is to **update all data structures** to indicate to each rescuing entity the ID of the entity in need of assistance. Furthermore, the drone also sets the state of all the entities involved in a **rescue state** on the grid so that they can no longer be seen by drones flying over the place. We decided to implement this functionality by defining a particular global entity state, called `'RESCUING'`, which models each of the entity being involved in a rescue process (either as responders or as victims). This prevents victims to be saved from different actors, which would likely result in a deadlock. The drones communicates to civilians through the aforementioned broadcast channel 'drone_signal', the civilian check if the message is for him and it will be civilians' duty to contact FRs based on whether they have been told to act as zero responders or to contact a FR.

4 Stochastic Models

We implemented also a SMC version of the NTA. The two actors that were modified are the Civilian and the drone.

4.1 Stochastic civilian

The civilian can decide not to listen to the drone's instruction, to model the fact that the civilian could be in panic or wouldn't trust the authorities but prefer to save himself. To reach this goal, we implemented two alternatives, depicted in Figure: 5 when the civilian receives a message on the `drone_signal` channel: with probability `Pr_Listen_CIV_probs[id]` the civilian will listen to the drone and reach the wait state like in the deterministic version of the timed automata. In the complementary case he will go back to the Active state and all the actions performed by the drone (i.e. the fact that the actors involved go into `RESCUE` state) will be reverted.

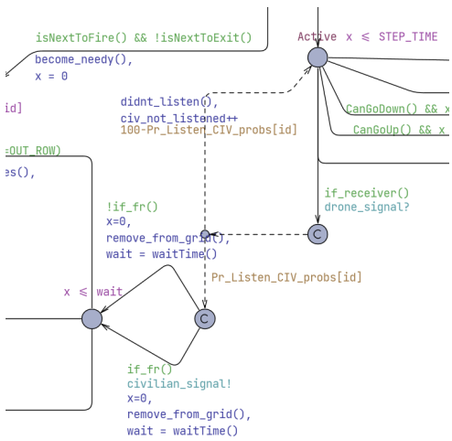

Figure 5: Stochastic civilian automata.

4.2 Stochastic drone

The drone's sensor can be faulty. This can happen before the search for civilian pairs happens. With probability `Pr_Fault_probs[id]` the drone's sensor doesn't work so the automata goes directly into the waiting state. Conversely if the sensor works the drone can begin scanning for civilians. We coun't the drone's fault only if the failure caused the undtetection of a pair of needy and not needy civilians. The picture of the automata is at Figure: 6

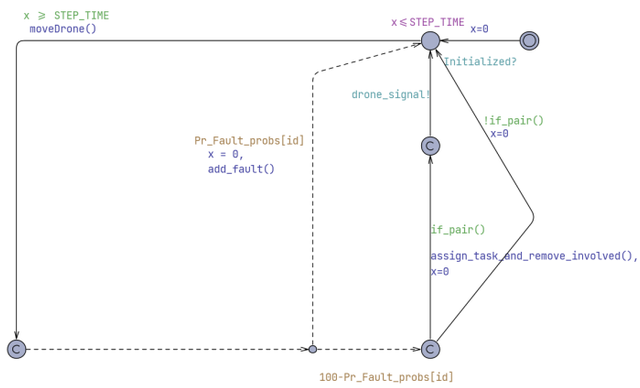

Figure 6: Stochastic drone automata.

5 Properties

Deterministic queries .
It is possible for a percentage N of all civilians to reach a safe state within time T;

$$(\varphi) : \exists\Diamond \left(\text{global} \leq T \wedge \left(\frac{\text{rescued\_civilians} \times 100}{\text{CIVILIAN\_NUMBER}} \geq N \right) \right)$$

Here we used the construct $\exists\Diamond(P)$ that signifies that P is reachable from the initial state. This can be rephrased in this context as: "there is a path from the initial state where global time is less than T and the percentage of saved civilians is greater than N".

A percentage N of all civilians is always guaranteed to reach a safe state within time T.

$$(\psi) : \forall\Diamond \left(\text{global} \leq T \wedge \left(\frac{\text{rescued\_civilians} \times 100}{\text{CIVILIAN\_NUMBER}} \geq N \right) \right)$$

Here we used the construct $\forall\Diamond(P)$ that signifies that P is inevitable. This can be rephrased in this context as "all paths from the initial state eventually reach a state where global time is less than T, and the percentage of saved civilians is greater than N".

Stochastic queries This query analyzes the probability that there is a path where the percentage of saved civilians is more than N within a time T.

$$Pr[\leq T] \left(\Diamond \left(\text{global} \leq T \wedge \left(\frac{\text{rescued\_civilians} \times 100}{\text{CIVILIAN\_NUMBER}} \geq N \right) \right) \right)$$

This query returns the confidence interval that always, at least a percentage of civilians, is saved within time T.

$$Pr[\leq T] \left(\Box \left(\text{global} \leq T \implies \left(\frac{\text{rescued\_civilians} \times 100}{\text{CIVILIAN\_NUMBER}} \geq N \right) \right) \right)$$

6 Settings analysis and results evaluation

6.1 First Layout

As the first environment to test, we decided on a square grid with dimensions set to 10. The fire has been positioned at the **center** to simulate some component failure in a work environment; civilians are spread onto the map. In particular, some civilians have been placed **close to the fire** (thus becoming a victim from the very first time instant). In contrast, some others have been placed along the two horizontal walls (North and South) to **simulate** a **tentative evasion from the smoke** (which would probably happen in reality). Exits, on the other hand, are on vertical walls (East and West) to simulate conditions such that civilians who find themselves endangered and those along the walls must try to reach the exits actively. Figure 7 depicts such configuration.

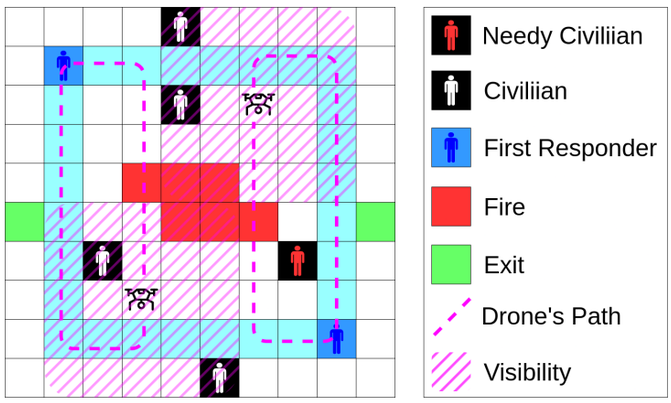

Figure 7: Setting 1

```
const int Tfrs[FIRST_RESPONDER_NUMBER] = {1,1};
const int Tzr[CIVILIAN_NUMBER] = {2,2,2,2,2};
const int Tv[CIVILIAN_NUMBER] = {5,4,4,5,6};
const int Nv[DRONE_NUM] = {2,2};
```

Listing 1: Array Initialization for Drones and Civilians

```
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 100))       ---> Satisfied
E<> (global <= 100 and (casualties*100/CIVILIAN_NUMBER >= 60))               ---> Satisfied
E<> (global <= 100 and (casualties*100/CIVILIAN_NUMBER >= 80))               ---> Not Satisfied
A<>(global <= 100 and (rescued_civilians * 100 / CIVILIAN_NUMBER >= 40))     ---> Satisfied
A<>(global <= 100 and (rescued_civilians * 100 / CIVILIAN_NUMBER >= 60))     ---> Not Satisfied
```

Listing 2: Queries for setting #1

First setting (shown in listing 1) From the reported queries 2 and the respective results, we can observe that in the provided map, there exists **at least one path in which all the civilians have reached a safe state**. However, this doesn't imply that this will occur in every possible scenario in such an environment. This poses the reason for further investigating the second and third queries that detect the threshold of the number of the number of civilians always saved. We can observe that, on the one hand, there exists a path along which 60 % of civilians die (thus confirming that the result of the first query is not globally accurate). Still, at the same time, we see that there is no path along which 80 % of civilians die (within a reasonable time bound of 100-time units). These last two queries are consistent with what we could conclude from the previous two, which let us suppose that, in such a configuration, for all possible scenarios, no more than two civilians (40 % of 5, the number of civilians) reach a safe state.

Second setting Let us now change some of the parameters in the configuration, as shown in listing 3, and investigate possible changes in the results.

```
const int Tzr[CIVILIAN_NUMBER] = {1,1,1,1,1};
const int Tv[CIVILIAN_NUMBER] = {7,7,7,7,7};
const int Nv[DRONE_NUM] = {4,4};
```

Listing 3: Second Array Initialization for Drones and Civilians

We discovered unexpected results by running the set of queries indicated in listing 4. Increasing the drone's visibility range (`Nv`) and the civilians' lifetime (`Tv`), while assuming zero responders can provide assistance more quickly, the results reported in listing 4 show a higher number of casualties and, correspondingly, a lower global number of rescued civilians—now only 20% (thus 1 out of 5 civilians). Nonetheless, the best case remains unchanged, as there still exists a path for which a state with 100% successful rescues is achievable. This leads us to the following consideration: In this setting, it is very likely that a civilian in danger is seen **by a drone before being noticed by a first responder**. In our model, if a civilian is instructed to help or to call a first responder, the endangered civilian will not be detected even if a first responder passes by. To make the model more realistic, it should be adjusted so that even if a drone instructs a civilian, the first responders can **still perform rescues**.

```
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 100))       ---> Satisfied
E<> (global <= 100 and (casualties*100/CIVILIAN_NUMBER >= 60))              ---> Satisfied
E<> (global <= 100 and (casualties*100/CIVILIAN_NUMBER >= 80))              ---> Satisfied
A<>(global <= 100 and (rescued_civilians * 100 / CIVILIAN_NUMBER >= 40))    ---> Not Satisfied
A<>(global <= 100 and (rescued_civilians * 100 / CIVILIAN_NUMBER >= 20))    ---> Satisfied
```

Listing 4: Queries for setting #2, second parametrization

First Layout stochastic model In this paragraph we present the analyis of the stochastic version proposed in the first setting.

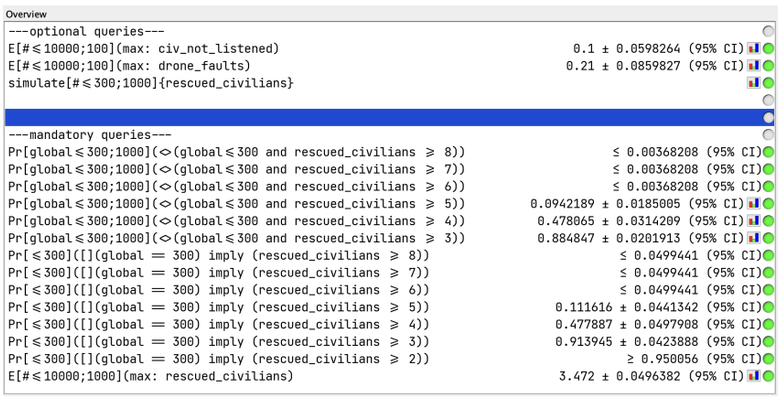

Figure 8: Stochastic queries for setting #1

Upon setting parameters as in listing 1 and in listing 5and performing the stochastic queries reported in 8 we obtain the following results. (NOTE: The time bounds have been set after experiencing convergence within 300 time units).

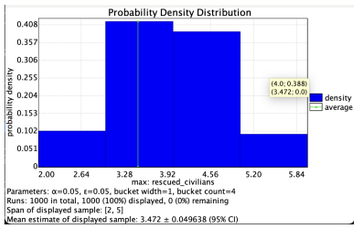

Figure 9: Visualization of the probability density distribution for the number of rescued civilians in setting #1

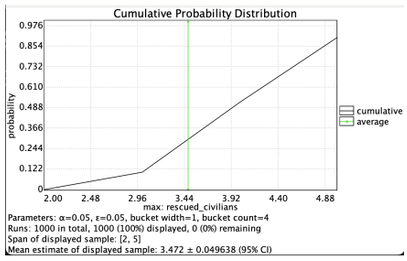

Figure 10: Probability cumulative distribution for the number of rescued civilians in setting #1

```
const int Pr_Listen_CIV_probs[CIVILIAN_NUMBER] = {70,75,70,80,75};
const int Pr_Fault_probs[DRONE_NUM] = {20, 25};
```

Listing 5: First probability initializations for stochastic setting #1

From these charts we can observe how the most likely number of rescued civilians for such a configuration is between 3.28 and 3.92, leading to an expected value of 3.472 as indicated by the query on the last line. In addition to that, from the series of queries we can see how the likelihood grows at the decrease of the number of rescued civilians queried. In particular, the turning point is when 4 civilians of the total 5 are saved. This is not only in line with the deterministic results, but also allows us to conclude that the possibility of 80% of civilians being saved is not remote, having roughly 50% probability. On the other hand we deduce that the likelihood of 100% civilians being rescued is low, as the first query (of the 'mandatory ones') yields a value of practically 0. This is proof of the importance of stochastic analysis, as it provides quantitative points of view.

```
const int Pr_Listen_CIV_probs[CIVILIAN_NUMBER] = {20,25,25,20,15};
const int Pr_Fault_probs[DRONE_NUM] = {80, 65};
```

Listing 6: Second probability initializations for stochastic setting #1

Let's now proceed with a different tuning of probabilistic parameters, as illustrated in listing 6. As a consequence of this parameter modification we can observe in figure 11 a coherent change in the number of drones' sensors' faults and of civilians ignoring the instructions. On the other hand, other measurements stay roughly the same due to the simple map configuration.

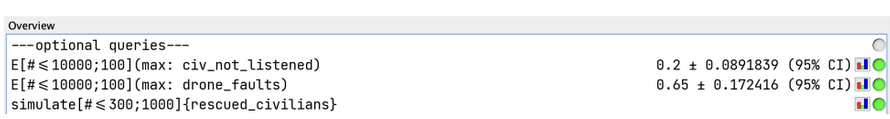

Figure 11: Number of civilians ignoring instructions and drones' sensors faults

6.2 Second Layout: a bigger map

We then tested the model on a square map of 14x14. The fire is centered but has a more irregular shape than the previous model (see Figure 12). We modelled the movement of the first responder in a circle around the fire, even though it will never reach into the coves because we thought that in a real life scenario this would be too dangerous since the first responder could be surrounded by fire. This causes the first responders not always to see the civilians in need because they could be far from the path that they are following. Also, in this case, we have civilians in direct contact with fire who will be injured from the first instant of the simulation, and we have civilians who are further out and, thus, should be able to reach the exits autonomously. The civilians not in direct contact with fire will be contacted by the drones if they enter their visibility range and drones detect a victim that needs to be rescued.

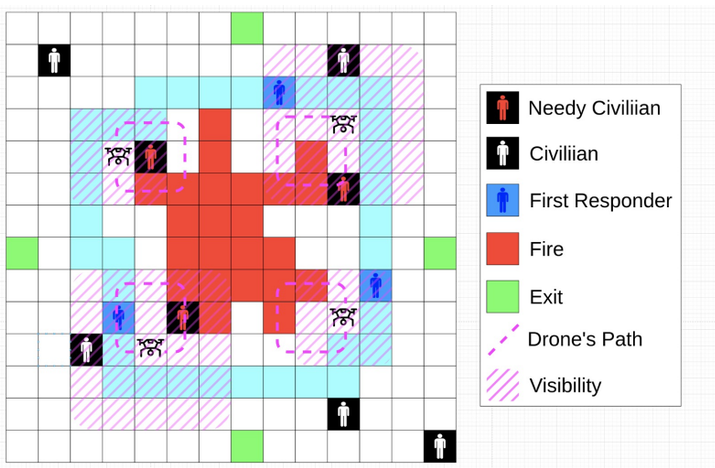

Figure 12: Setting 2

```
const int Tfrs[FIRST_RESPONDER_NUMBER] = {1,1,1};
const int Tzr[CIVILIAN_NUMBER] = {3,2,2,4,2,3,3,2};
const int Tv[CIVILIAN_NUMBER] = {4,2,5,3,2,6,3,1};
const int Nv[DRONE_NUM] = {2,1,2,1};
```

Listing 7: First array Initialization for Drones and Civilians

First setting By initializing the parameters like in listing 7, we aim to show how the drones can successfully help the first responders detect civilians in case they cannot go too close to the fire. From the query results, we can see paths where the percentage of saved civilians is at least 80%; however, both the 90% and 100% queries fail. We can detect from this analysis that the break-even point is at 7 out of 8 saved civilians, meaning that there will always be at least one casualty. In fact even though civilian in position (4,4) could be saved directly by first responders, he will die before their arrival since, at first, two of them are already busy helping the civilians found by the drone.

Furthermore, by analyzing the queries meant to verify *property 2*, we can see how at least 60% of the civilians are saved among all possible paths. This means that 5 out of 8 civilians will always be saved. Moreover, the fact that the query with the 65% is not satisfied shows that despite the fact that there exists a path with 80% of civilians that are saved, this condition does not hold on every path.

```
const int Nv[DRONE_NUM] = {1,1,1,1};
```

Listing 9: Second Initialization for Drones and Civilians

```
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 80))    ---> Not Satisfied
```

Listing 10: Results for setting #2, second parametrization

Second setting By keeping all the parameters the same but reducing the visibility range of the drones to 1 we can see how the rescue is less effective. The query that searched for the existence of paths where at least the 80% of civilians is saved now fails, meaning that with the old parameters there was at at least one scenario where the 7th civilian could be saved, while here this is not true.

```
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 70))    ---> Satisfied
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 80))    ---> Satisfied
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 90))    ---> Not Satisfied
E<> (global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 100))   ---> Not Satisfied
A<>(global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 20))     ---> Satisfied
A<>(global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 60))     ---> Satisfied
A<>(global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 65))     ---> Not Satisfied
A<>(global <= 100 and (rescued_civilians*100/CIVILIAN_NUMBER >= 80))     ---> Not Satisfied
A[] (not deadlock)                                                      ---> Satisfied
```

Listing 8: Results for setting #2, first parametrization

Second Layout stochastic model Here the analysis has been performed on the stochastic model. The parameters are the same of the first deterministic version, while the probabilities of fault of each drone's sensor and the probabilities of each civilian to listen to drones' commands are displayed in listing 11.

A first look of the behaviour of the number of civilians saved tells us that after 300 seconds we can be pretty sure that the simulation converges, as shown in figure: 14. For this reason we choose to verify the queries with this time bound.

```
const int Pr_Listen_CIV_probs[CIVILIAN_NUMBER] = {70, 80, 60, 80,70,90,80,80};
const int Pr_Fault_probs[DRONE_NUM] = {20, 30,30,40};
```

Listing 11: Parameters

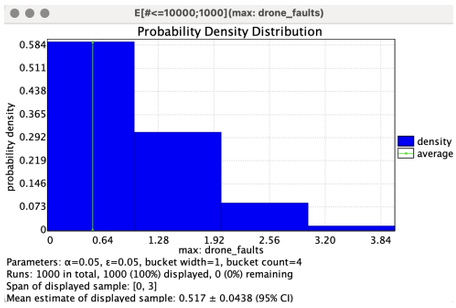

Figure 13: Visualization of the probability of the maximum number of drones faults

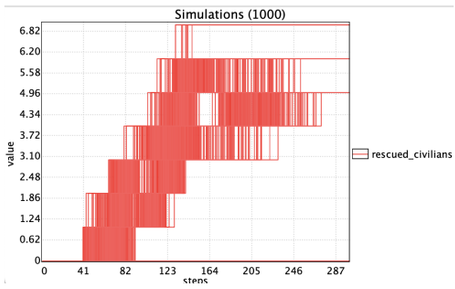

Figure 14: Setting 2 stochastic simulation of the number of civilians saved

Figure 15: Results of the first stochastic query

Figure 16: Results of the second stochastic query

From this analysis of the queries in figure 15 and 16 we can infer that it is highly probable even in the scenario with faulty drone sensors and civilians that may ignore instructions that at least 5 civilians are saved. This is probably due to the fact that those civilians are saved by fist responders or they autonomously reach the exits. Meanwhile, we can see that the probability of saving 6 civilians halves, and it is highly unlikely to save 7 civilians and it is almost impossible to save all of them, as the reported confidence interval is practically a constant 0. By investigating figure 13 we can observe that the probability distribution reaches its maximum value for a smaller number of drones' sensors' faults (as expected) and decreases with an exponential behaviour. This perfectly represents real-world system, which are typically modelled with exponential distributions (or derivations of such, for instance: Γ distribution, *Weibull* distribution). Overall, the average probability for a drone's sensor to be faulty is 0.517 as shown in figure 13.

7 Conclusions

To summarize, this search and rescue model has been built with the goal of balancing realism and complexity. We tried to model the communication with the various actors using channels and global variable to simulate the exchange of information and we implemented movement policies that in our view could be reasonable in the case of a real life scenario, like the movement of the first responder around the fire (that is the place where hurt civilians will be found), by making the drones hover over the fire, to have a better change of seeing injured civilians and by making the civilians move toward the closest exit. Most of the effort has been spent on simplifying the model so that the query could run in a reasonable amount of time. The configuration presented showed how parameters change the way the model behaves and in particular how the drones are useful in the rescuing process.