POLITECNICO MILANO 1863

Formal Methods for Concurrent and Real-Time Systems

FORMAL ANALYSIS OF SEARCH-AND-RESCUE SCENARIOS

Homework Project

Presented to: Prof. Pierluigi San Pietro & Dr. Livia Lestingi

By:

Manuela Marenghi 10716532,

Simone Lucca 10761586,

Louis Dutheil 11028762

Spring 2024

Abstract:

This report presents a UPPAAL model for a scenario of search and rescue. The global idea is to represent a scenario of search and rescue with civilians that need to be saved from a dangerous environment with fire, first responder trying to save the civilians and drones dispatching instructions to the civilians.

CONTENTS

1 DESIGN

1.1 Area layout

The grid is designed as a rectangular layout with the specific constraint the number of drones must not exceed the number of columns. In the model, it is crucial that no two drones occupy the same cell simultaneously. Therefore, if the number of columns exceeds the number of drones, it ensures that each drone has a distinct group of columns to operate in, thus preventing any overlap.

Drone positioning is determined by their IDs, with the drone having the lowest ID positioned on the far left of the grid, and subsequent drones placed in progressively rightward columns. Drones can be positioned in the middle of a column, only if it is their own. If a drone is not correctly positioned, it will be reset to the top-left cell of its designated column.

Exits are situated at the edges of the grid and serve as escape points for civilians. The model reflects the civilians' instinct to reach safety by moving towards the nearest grid edge, where exits are most likely to be found. Additionally, civilians are modeled to avoid fire, ensuring they do not move towards cells adjacent to a fire.

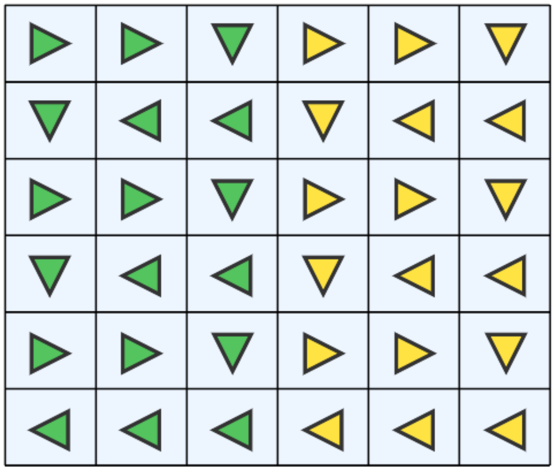

Figure 1. Illustration of the drones' movements

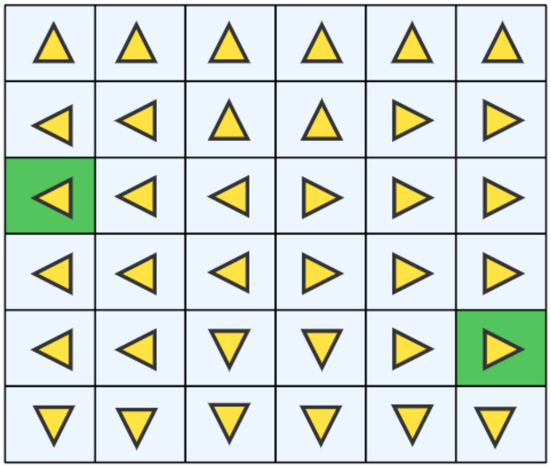

Figure 2. Illustration of the civilians' movements gradient

Figure 1 illustrates the movement of two drones within a 6x6 grid. The drones follow their designated paths, ensuring that no two drones occupy the same cell simultaneously. Figure 2 depicts the movement gradient of civilians within the same grid. This gradient reflects the civilians' instinctual behavior to move towards the nearest edge in their attempt to reach safety, assuming the worst-case scenario where visibility is zero. The exits, where civilians can escape to safety, are represented in green.

1.2 Stochastic features

The following stochastic features are modeled in Homework_stocv.xml, where an uncontrollable edge with a probabilistic weight is added:

- The drones' sensor failure rates, represented by the variable *Pf* (probability of failure), which is a characteristic specific to each drone.

- The humans' instruction acknowledgment rates, represented by the variable *Pr* (probability of accepting to help a person in need), which is a characteristic unique to each instantiated civilian. This probability applies each time a civilian receives information from a drone, whether they are instructed directly to help or instructed to call a first responder.

1.3 Design assumptions

1. No two entities, between civilian and first responder, can be found on the same cell of the grid.
2. No entities can be found on cells with fire or exits since fire will immediately result in their demise, and any person located on an exit will have already left and be considered saved a priori.
3. Distances on the grid are always measured using the Manhattan distance. This distance metric calculates the total number of horizontal and vertical steps required to move between two points, providing a straightforward and practical measure for movement within the grid.
4. Survivors will feel morally obligated to assist people in need and will always follow the instructions provided by the drone.

2 MODEL DESCRIPTION

2.1 High Level Description

Our model is composed by four entities:

- Civilian
- Drone
- First Responder

Civilian

The civilians must be saved, but they are not passive, they save themselves by leaving using an exit, or they can even save / help other civilians when instructed by a drone.

Drone

The drone's mission is to patrol over the grid and when detecting a civilian in need of assistance it must communicate its presence to another civilian around so that this one can help.

First Responder

First Responders navigate the grid with the primary objective of assisting individuals in need. They will provide help if they are near a person requiring assistance or if they are directed by a civilian who has been instructed by a drone to seek their aid.

2.2 Component description

2.2.1 Civilian

After initiation, the current grid positions of the civilians are evaluated for safety. If a civilian is in a safe position (with no fire within one unit of distance), they are considered a SURVIVOR. If not, the civilian is classified as InNeed and will require assistance. If they do not receive assistance after a defined lapse of time they die and are removed from the grid. If a civilian is located near an exit, they are considered saved and are removed from the grid.

In Need:

- Once a civilian enters the "*InNeed*" state, they will remain in that condition. If they are not assisted within *Tv* time units, they will perish, emitting a signal on the "dead" channel to indicate their demise.

Survivor:

- A survivor prioritizes saving their own life by moving towards the nearest edge of the grid, anticipating that an exit will be located nearby. But when instructed by a drone, the survivor will be obliged to execute the drone's instructions that can be of helping directly a person in need (*ZeroResponder* state) or calling a first responder to help the person in need (*CallingFirstResponder* state).

ZeroResponder:

- The survivor will stay in this state, without changing its own position in the grid, for as time as it is needed for it to reach the person in need, once reached it will need *Tzr* time units to save the person in need. If during this process, the person in need dies, then the zero responder will turn back to being a survivor, otherwise both the zero responder and the person in need will be saved.

CallingFirstResponder:

- The survivor will call the first responder located in the cell indicated by the drone, using a shared variable. This process involves waiting for a duration equivalent to the distance between the survivor and the first responder. Once this time elapses, the survivor will notify the first responder about the person in need of assistance. The first responder will then proceed with the helping procedure. If the assistance is successful, both the survivor and the person in need will be considered saved. If not, the survivor will revert to the survivor state.

2.1.2 Drone

After being initialized the drone moves until a PNA (person needing assistance) and a survivor is detected within *Nv* distance units from itself.

In the scenario depicted, the drone has a detection range of 3 cells. Within this range, the drone can detect and contact a nearby civilian (zero responder) to assist a person in need who is near a fire. If a first responder is present, the drone will instruct the zero responder to contact the first responder. This decision is made to ensure the presence of the first responder before involving the civilian, thereby reducing the risk of the civilian failing to find help. If there is no first responder nearby, the drone will instruct the civilian to help the person in need directly.

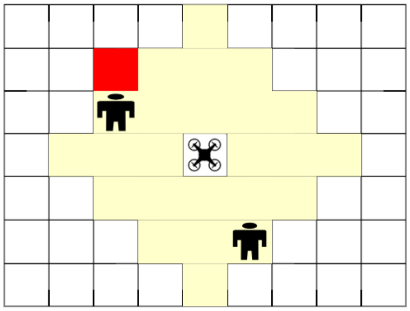

Figure 3. Drone instructing survivor to help PNA

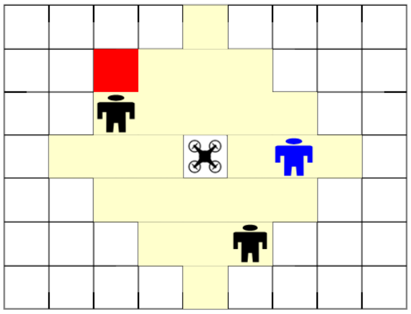

Figure 4. Drone instructing survivor to call a first responder

2.1.3 First Responder

Following its initialization the First Responder has three courses of action:

1. If a civilian in need of assistance is detected nearby, the first responder will rescue the civilian
2. If called by a survivor, the first responder will travel to rescue the civilian in need of assistance.
3. If none of these scenarios happen, the first responder simply moves toward the center of the grid

Civilian in need of assistance detected:

The first responder is registered as busy and the coordinates of the civilian they are helping are registered using `saveIdRescuing()`. An intermediate check to see if the civilian in need is still alive is realized, if dead the first responder is freed using `setFRFree()` and goes back to the initial state. Same actions are taken after Tfr time units to complete the rescue.

Called by a survivor:

After being called by a survivor through the channel `assistDone`, the time to reach the civilian in need of assistance is calculated using `setWaiting()`. The first responder spends time reaching the civilian in need and then spends time performing the rescue. An intermediate state is used to verify if the civilian in need is still alive if not the rescue mission is aborted. If the rescue is completed the first responder is freed.

3 SYSTEM CONFIGURATION

3.1 System parameters

Table 1. System parameters and their descriptions

Parameter	Description
GRID_W	Grid width
GRID_H	Grid height
N_CIVILIANS	Total number of civilians
N_DRONES	Total number of drones
N_FR	Total number of first responders
Tv	Time for a civilian to die close to the fire
global_time	Absolute time of the system
grid[GRID_W][GRID_H]	Matrix representing the scenario
pos[N_CIVILIANS+N_FR]	Array of coordinates of all Civilians and First Responders

posDrone[N_DRONES]	Array of coordinates of Drones
helping[N_CIVILIANS]	Array of coordinates of persons assisted by Zero Responder of id equals to the position in the array
calling[N_CIVILIANS]	Array of coordinates of First Responders contacted by Zero Responder of id equals to the position in the array
FRHelping[N_FR]	Array of coordinates of persons assisted by First Responder of id equals to the position in the array plus N_CIVILIANS
n_safe	Number of Civilians saved
Tscs	Time to evaluate properties
N_perc	Percentage of Civilians safe to evaluate in the verifier part

3.2 Channels

To synchronize entities, we define the following communication channels:

- `starting` is the channel used by all the entities to start at the right time, the signal is emitted by the Initializer
- `assistDone` is used to ensure synchronization between survivor and first responders/people in need of assistance.
- `dead` is used to signal a person in need of assistance has died and the rescue mission can be canceled
- `callFirstResponder` is used by the drone to tell the nearby survivor to contact a first responder
- `assist` is used by the drone to tell the nearby survivor to assist the person in need of assistance

3.3 System set up

When creating a Scenario is fundamental to insert the grid configuration in the *initializeGrid()* function in the *'Declarations'* file, following these rules:

1. Each entity is represented by an enumerated variable in the *Declarations' file.* To set fire and exits cells it's sufficient to indicate as follows:
   e.g. grid[y][x] = FIRE
2. To place entities, there is a specific function *placeEntity(id, Y, X, entityType)* as follows:
   e.g. placeEntity(0, 0, 1, CIVILIAN)
3. At the end of the function the following initialization must occur *initCallingHelping(), initFRHelping()* and *n_safe=0*.
4. Last configuration must be in the *'System Declarations'* file where it is possible to indicate for Civilian its *Tzr,* First Responder its *Tfr* and initialization of a value idRescuing, drones *Nv.* In the stochastic version it is also possible to indicate *Pr* (probability of giving assistance) and *Pf* (probability of failure) respectively for Civilians and Drones:
   e.g. C0:= Civilian(0, 3) where 0 its the id

4 TESTING SCENARIOS

4.1 Grid layout

Three layouts were created for testing purposes, based on potential real-world scenarios. The proposed test cases can run on a standard PC within a reasonable amount of time. However, for more realistic scenarios, the model would require more powerful computational resources.

4.1.1 Test 1

This test case is meant to cover the case of corridor on fire, the exit on the right could be an emergency exit or a window with firemen that assist people from the outside.

4.1.2 Test 2

This test case covers a scenario where the number of first responders is enough to help civilians in need.

4.1.3 Test 3

This test case is meant to cover the case of a more complex situation to stress the system with a bigger grid and two drones. In this case there is only one first responder so collaboration from survivor is necessary to help civilians in need.

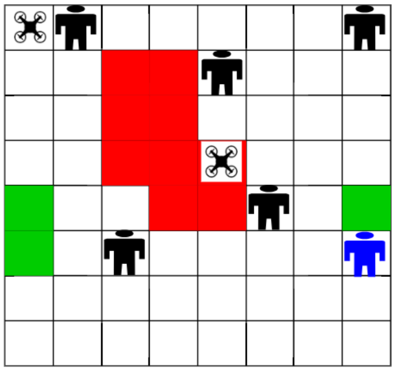

Figure 5. Grid layout 3

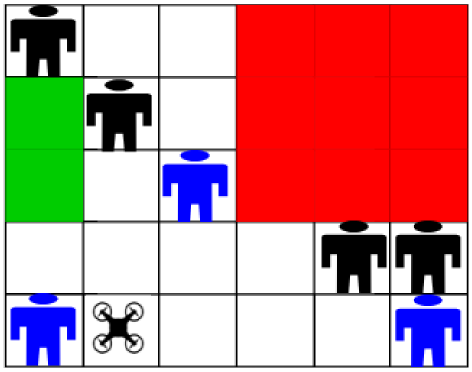

Figure 6. Grid layout 2

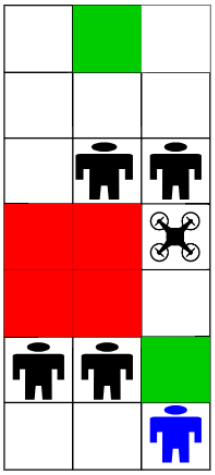

Figure 7. Grid layout 1

4.2 Testing parameters

Matching these three different grid layouts, we have three sets of parameters for these simulations. The parameters are listed in the table below.

Table 2. Simulation parameters used to test the different scenarios

	Grid size	N_CIVILIANS	N_DRONES	N_FR	Tv
Scenario 1	3x7	4	1	2	13
Scenario 2	6x5	4	1	3	4
Scenario 3	8x8	5	2	1	10

5 PROPERTIES

5.1 Deterministic version

5.1.1 Property 1

This property checks if there exists a path where within the specified time *Tscs*, at least N% of all civilians have reached a safe state. This means that it is possible for the system to ensure that a certain percentage (N%) of civilians are in a safe state within the given timeframe. This query is used to verify the potential for achieving a safety condition within a specified time limit.

QUERY: E<> (global_time <= Tscs and n_safe >= (N_CIVILIANS * N_perc / 100))

5.1.2 Property 2

This property checks if a percentage *N_perc* of all civilians is always guaranteed to reach a safe state within a specified time Tscs. This means that for every possible execution path, once the global time reaches or exceeds Tscs, at least *N_perc* of the total number of civilians (*N CIVILIANS*) must be in a safe state. This query ensures a minimum level of safety within the given timeframe.

QUERY: E[] (global_time >= Tscs imply n_safe >= (N_CIVILIANS * N_perc / 100))

5.1.3 Property 3

This query checks that in all possible states of the system, whenever the global time reaches or exceeds Tscs, at least *N_perc* of the total number of civilians are guaranteed to be in the saved state. It ensures that the system meets this requirement under all circumstances.

QUERY: A[] (global_time >= Tscs imply n_safe >= (N_CIVILIANS * N_perc / 100))

5.2 Stochastic version

In the stochastic version we've changed the structure of query 1 and 3, mantaining their meaning, to make them suitable for *Pr[]* operator. We also add a third query to quantify the expected value of the variable *n_safe*:

5.2.1 Property 1

This property checks the probability for which, in a specified number of time units, there exists a path where within the specified time *Tscs*, at least N% of all civilians have reached a safe state. The meaning is the same for the deterministic version, it only changes its expression as we can use the operator *Pr[]* to express the requested time *Tscs.* In the project it is possible to observe two versions of this query as they're used for different scenarios (specified in the comments).

QUERY: Pr [Tscs] (<> (n_safe >= (N_CIVILIANS * N_perc / 100)))

5.2.2 Property 2

This property checks the probability for which, in a specified number of time units, all possible states of the system, whenever the global time reaches or exceeds Tscs, at least *N_perc* of the total number of civilians are guaranteed to be in the saved state. In this case we've decided to use an extreme amount of time, to make it clear that after the evolution of the scenario is ended, *n_safe* doesn't change:

QUERY: Pr[<=10000] ([] (reached_Tscs() imply n_safe >= (N_CIVILIANS * N_perc / 100)))

5.2.3 Property 3

This property checks only the expected value of the variable *n_safe*, that is used to keep track of civilians safe over time units in limited number of runs:

QUERY: E[<=1000;10] (max: n_safe)

6 VERIFICATION RESULTS

6.1 Deterministic version

6.1.1 Test parameters

The parameters related to the testing condition are the following: *Tscs* represents the amount of time in which a percentage N% of civilians is able to become safe.

Table 3. Test parameter used for the different scenarios

	Tscs	**N%**
Scenario 1	10	30
Scenario 2	5	40
Scenario 3	5	20

6.1.2 Test results

Table 4. results of the three queries with the three scenarios

	Query 1	**Query 2**	**Query 3**
Scenario 1	Satisfied	Satisfied	Satisfied
Scenario 2	Satisfied	Satisfied	Satisfied
Scenario 3	Satisfied	Satisfied	Unsatisfied

As shown in Table 4, almost all queries were satisfied, though some required longer validation times. For instance, in Scenario 2, the third query took nearly 10 minutes to validate. Notably, the third query in Scenario 3 was not satisfied. This failure is attributed to the possibility of the first responder moving in the wrong direction due to lack of visibility in the room. Consequently, no person in need would be saved, resulting in the failure of the third query.

6.2 Stochastic version

For this version we've decided to create 3 scenarios as follows:

- Scenario 1-2: the grid configuration (entities, fire, exits) remains the same but with changed probabilities specified. In the first scenario we have a “bad configuration” of the system, having low probabilities of following instructions for Survivor and high probabilities of sensor's failure for drones. The second scenario represents a better version in terms of these probabilities, having Survivor with high probabilities of following instructions and drones with low sensor's failure.

- Scenario 3: in this configuration we've tried to stress the system by increasing the size of the grid. Also, this configuration is bad in terms of probabilities.

6.2.1 Test parameters

Table 5. Test parameter used for the different scenarios in the stochastic version

	Tscs	N%
Scenario 1	35	50
Scenario 2	35	50
Scenario 3	19	30

6.2.2 Test results

Every test is made by maintaining a 95% level of confidence, meaning that there is a 95% chance that the results obtained from the test will fall within the true population parameter.

Scenario 1-2:

To better analyze results obtained it's useful to compare these two scenarios to see if different probabilities lead to different behaviors of the same grid.

Starting from the first query it's useful to observe the *Cumulative Probability Confidence Intervals* graph of the two scenarios. The graph provides a visual representation of how the cumulative probability of the query evolves over time along with the confidence intervals. It's possible to see that results don't change too much; we only get lower results during t <= *Tv*. This is because the specific query isn't very restrictive as it only needs an existing path. The peak observable near the end of runs duration it's because in our system design Survivor are considered safe at the end of *Tv* time, when the scenario is ended, or after an helping action:

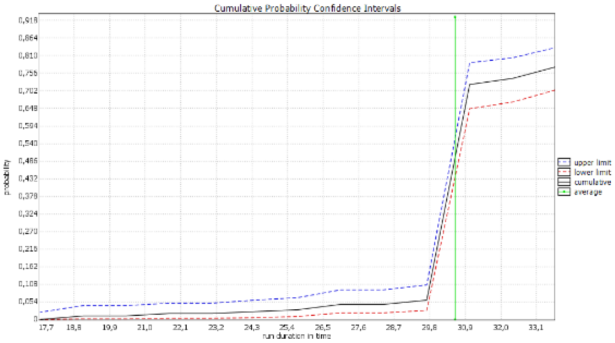

Figure 8. Cumulative Probability Confidence Intervals graph of the Scenario 1

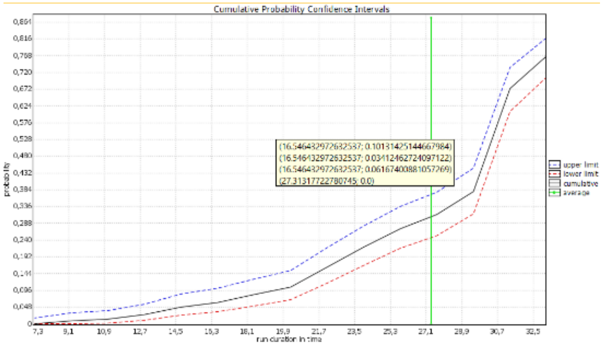

Figure 9. Cumulative Probability Confidence Intervals graph of the scenario 2

In the second verification it's possible to observe the main difference between the two scenarios. In fact, this query search among all paths making it more restrictive. Scenario 1, that has low probability of having civilians helped by other civilians, relies only on first responder behavior. Scenario 2 has more possibilities of having civilians rescued:

Table 6. Verification results of query 2 in both scenarios

	Probability
Scenario 1	0.0599014 ± 0.0397112
Scenario 2	0.410671 ± 0.0492968

Scenario 1 has an almost zero probability of satisfying this property, whether Scenario 2 has it more significant.

It is interesting to observe the results of query 3 to understand the stability of Scenario 2 compared to Scenario 1. The overall results for both scenarios are similar. However, examining the graph over limited runs reveals that in Scenario 2, the results remain more consistent across different runs, whereas in Scenario 1, the results rapidly decrease:

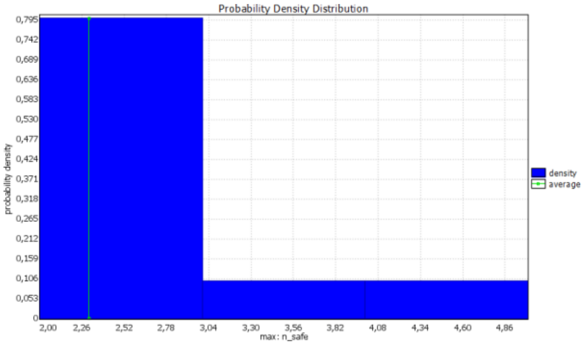

Figure 1. Scenario 1

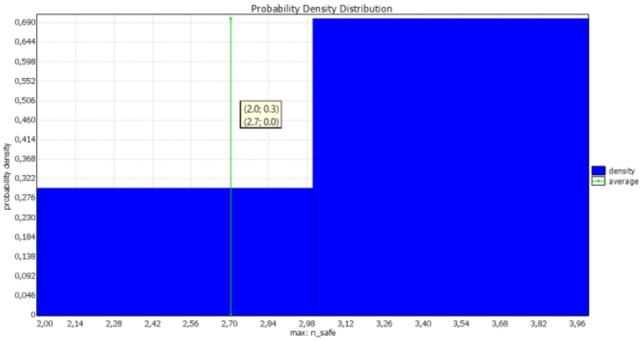

Figure 2. Scenario 2

Scenario 3:

This scenario represents a more complex case in terms of grid configurations, characterized by a low probability of civilians acknowledging instructions from drones and a high probability of drone sensor failures. In this context, *Tscs* is set to be less than *Tv* to verify properties before the scenario concludes. As a result, given that for this specific *Tscs* we are only considering civilians saved after being in need and the configuration scenario, it is possible to observe low probabilities for both queries:

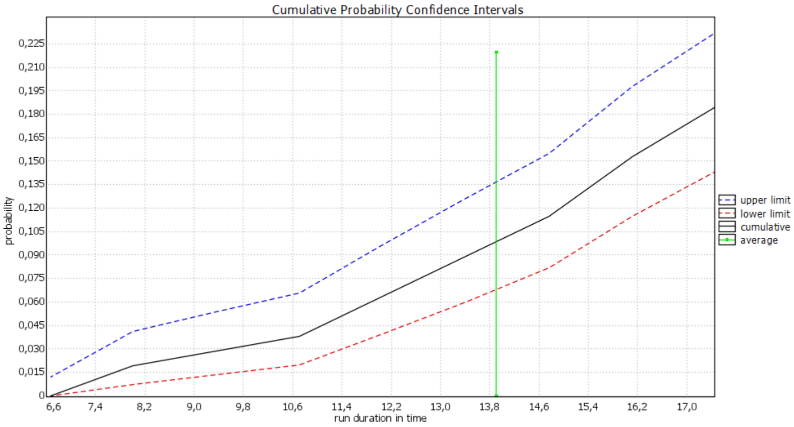

Figure 1. Property 1 graph

Property 2 also has a low probability of meeting the parameters, with a result of 0.220671 ± 0.0472652. In this case, both properties exhibit similar low probabilities

7 CONCLUSIONS

The model was created based on specific assumptions and should not be considered definitive for real-world scenarios, as the behavior of civilians and first responders can vary due to unpredictable external events. However, this model is valuable for evaluating and optimizing the layout of this type of security systems. By adjusting the positions of fire, civilians, and first responders, the model can help ensure that the overall topology is effective and robust.