Emergency Responses

André Leitão
Instituto Superior Técnico
Av. Rovisco Pais 1
+351 963463057
andre.filipe.afl@gmail.com

João Tavares
Instituto Superior Técnico
Av. Rovisco Pais 1
+351 925258962
joaobernardo.28@gmail.com

Madalena Pedreira
Instituto Superior Técnico
Av. Rovisco Pais 1
+351 917620100
mena.pedreira@gmail.com

ABSTRACT

Multi-agent systems have an enormous range of applications, from optimization to learning problems. In the context of the Autonomous and Multi-Agent Systems course at Instituto Superior Técnico, we decided to implement a multi-agent system responsible to respond to emergency situations. These (multi)agents are to effectively distribute ambulances and resources to efficiently respond in accordance with the type, gravity, amount of energy fuel (of the ambulances) and spatiotemporal distribution of emergency requests.

Keywords

Reasoning, Communication, Coordination, Search, Planning, Interaction, Agent-Oriented Programming, Resources, Decision Making.

1. INTRODUCTION

Our multi-agent system procures to understand the most efficient response mechanism to deliver the best performant emergency response system - it revises the traditional emergency response system's approach and procures a better way to deal with emergency requests.

In order to do so, we developed two distinct core types of simulations. In this report, we will define each of them, describe the metrics used to assess their performance and evaluate them according to those same metrics.

The simulation definition will follow two steps: firstly, determining the common characteristics of the simulations - the intervening agents, the environment and emergency conditions - and, secondly, by describing the different response approaches. An additional information about the system's display information was also found to be relevant for the better understanding of its execution.

Lastly, the evaluation phase will serve to ascertain the most performant type of approach, discussing the most relevant aspects of the simulation.

2. PROBLEM

Every society has its own way of dealing with its population's emergency needs. Consequently, emergency response efficiency has been a major concern to every civilization. Its performance is a focus point of debates as the years go on and, as innovative technology or better health paradigms are introduced, is constantly being reassessed.

Despite these technological breakthroughs, improving an emergency system is a fairly complex problem. Its complexity derives from the fact these systems are dealing with a great set of different variables that interconnect with one another, merging only by having the same goal: to answer to emergency situations in the best possible way.

2.1 Environment

The first stage of the system's modelling is to define its environment. We represent our environment as a network, an undirected graph, where each node corresponds to either a hospital, a patient or a charging station; each of its edges has an associated weight, representing the amount of energy that an ambulance needs to traverse those locations - this discount factor is a direct proportional value between distance of the nodes.

Formally, we can define our environment with the following properties: *Accessible*, since the agents can obtain complete perceptions of the environment's state; *Deterministic*, considering that all actions have guaranteed effects; *Dynamic*, considering that the environment is constantly being updated with the appearance of new patients; *Non-episodic*, since each emergency request and action happens in a specific time step and depends on the previously chosen decisions and *Discrete*, considering that the actions and perceptions are of a finite universe and depend solely on the agents.

2.2 Agents

Considering we will be dealing with emergency medical services in our project, we define the following agents: ambulances, hospital, charging stations and patients. These will intervene within the systems' environment and their collaboration will result in the outcome of the response.

2.3 Requirements

Our system has strict requirements to fulfil, as the contexts of its performance has great values at stake. We made an inventory of the primordial importance requirements and categorized them, adapting our system to convey these standards.

Firstly, emergency patients have a priority associated with their emergency needs. This emergency type request has a specific time period for the ambulances to respond to it, in accordance with the gravity of the situation. Consequently, *ambulances should pick up the patients before their priority time expires*.

Secondly, the emergency transportations are electric and start off with a full percentage of energy. As ambulances move through the environment, they are discounted in this resource, imposing a restriction on the availability of the ambulance as they need to *have enough energy to respond to the emergency situations*, that is, enough energy to move to the patient's location, picking it up, and dropping it in the hospital.

Finally, the efficiency of emergency responses relies on how fast emergency responses are. For that reason, the system should strive to achieve the best performance possible within the environment and energy constraints. That is, *aiming to arrive at the emergency scenario as quickly as possible*.

3. SOLUTION

For our solution, we decided on an object-oriented Python implementation where each agent is represented as a class, wherein internal state is represented by attributes, and sensors and actuators by methods.

The emergency scenarios were then simulated by the use of a network structure intended to convey the environment in which such agents interacted.

The different environments and scenarios tried to either illustrate a more traditional approach to the one seen in real-life emergency responses or an optimized version, evaluating the outcome of different agent's decisions in such scenarios.

3.1 Network

Initially, we contemplated the usage of a matrix to represent the agent's environment. After some consideration we revoked that idea and decided to embody the environment through a network representation. The graph has the same predicted behavior: the agent's position is a specific node and each agent's movement within the environment has a cost (for the network has weighted edges), discounted in its internal state (the ambulance's energy fuel).

We used the python library Networkx in order to set up the network structured environment for our agents to engage in. The library realizes a simplistic approach to illustrate the agent's positioning and trivial parameterization of the overall environment's structure - it encapsulates the desired functionalities and environment's attributes.

Consequently, our system uses the number of nodes parameterization, the edges weight range and the network (graph) type (we used a binomial graph) to depict the environment of our system. Figure 1 illustrates a generated graph with the default parameters. The blue node is the hospital and the green nodes are charging stations.

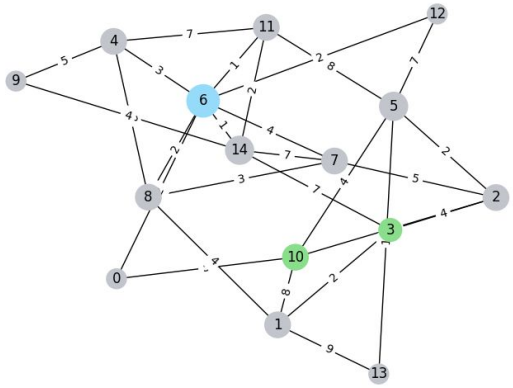

Figure 1: Example of a randomly generated network

3.2 Ambulances

We've followed the requirement's specification discussed in our project proposal. Concretely, this agent relies on a *reactive* architecture, it is *deliberative*, has a *belief-desire-intention* model, is *autonomous* and *adaptive* to its goals, possesses *rational* quality, maintains an *internal state* and presents *mobility* in the environment. In addition, for part of our simulations, it also has a *sociable* and *collaborative* behavior.

As for the ambulance's internal state, the agent has a *goal* (whose possible values are "no_goals_yet", "pick_up_patient", "go_to_hospital", "go_to_energy" and "get_energy"; an additional "head_position" goal is also required for learning purposes). Figure 2 describes how the agent's goals shift based on the environments and agent decisions:

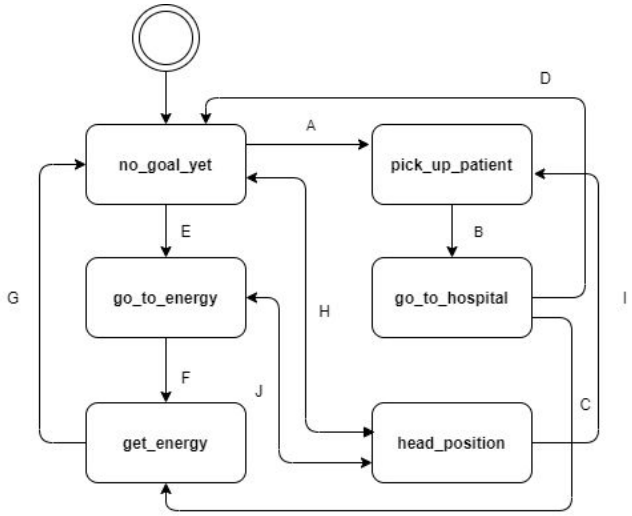

Figure 2: State machine of the ambulance's goals

As for the transitions between each goal's state, the agent follows these rules:

A - Starting point goal to every ambulance. Once an ambulance has a patient assigned, its goal switches to pick_up_patient.

B - Once the ambulance arrives at the patient's position (has the patient) its goal switches to go_to_hospital.

C - Once the ambulance leaves the patient at the hospital, if its energy is below 50%, it charges its battery in the hospital, switching its goal to get_energy.

D - Once the ambulance leaves the patient at the hospital, if its energy is above 50%, its availability to other agents switches, marking the goal no_goal_yet.

E - If an ambulance doesn't have any goal, and its battery is below 50%, it picks a charging station, beginning to take its path to it, switching the goal to go_to_ernergy

F - Once an ambulance arrives at its assigned charging station, the ambulance switches its state to get_energy. Marking the event that the charging station is beginning to charge the ambulance.

G - Once an ambulance arrives at its assigned charging station, the ambulance switches its state to get_energy. Marking the event that the charging station is beginning to charge the ambulance.

H - As long as the ambulances possess reinforcement learning, they select a position from their neighbors (adjacent nodes) to go there, setting the goal to head_position. Once they arrive there, the goal is reset to no_goal_yet.

I - Whenever an ambulance has a patient to pick up, even if it is heading a new position, the ambulance changes its priority to the patient.

J - Once an ambulance arrives at some place, if its energy is below 50%, it charges its battery at the nearest charging station. After the ambulance gets the battery fully charged, it switches its goal to heading_position.

Additionally, it has an *available* boolean - indicating if its goal is is flexible enough for it to change (the ambulance's availability for other desires depends on the goal it is pursuing, for example, if the agent is charging and energy above 70%, it could interrupt its activity to pick up a patient).The ambulance also has the agent's *path* to the goal, an id, the ambulance's *position* in the graph and its *available energy*.

The ambulance's positioning varies for each simulation's approach. In the traditional, *Hospital-Deliberative* approach, every ambulance starts off in the hospital - the ambulance's starting position coincides with the hospital one. For the learning collaborative ambulance's simulation, this agent's positioning varies - the ambulance's position in the time an emergency request is made could be any state in the system, it is what the agent learnt was best at.

It's energy fuel value is 100, indicating the percentage of the battery's capacity. This internal state is constantly being checked as it is the main restriction of the overall capacity of the ambulance to achieve its goals - as it needs to have enough fuel to respond to emergencies.

Pondering on the capacity of the ambulance, another candidate of attributes for this agent, we believed it would be best to restrict its fitness to *only one patien*t per time. The overall complexity of having more than one patient to respond to would not compensate as it would limit our conclusions of the other parameters modifications.

Once describing the agent's actuators, we can enumerate: responding to an emergency (going to where the patient is), picking up the patient, delivering the patient to the hospital, going to the energy charging station, and explore the locations on the map, namely, select a new location to drive, according to its *action selection* result. These basic functions are triggered in accordance with the agent's goal changes. For example, once the agent's goal switches to "go_to_hospital", the actuator "delivering patient to hospital" is actioned.

In a collaborative environment (our second simulation), the ambulance also needs to have the coordinates of the hospital, as well as for the other ambulance's decision to pick up the patient.

Any competition simulation in this environment would not be coherent with the system's context - competing over which ambulance reaches a patient could deprecate an emergency response, endangering a patient - therefore, throughout our system's implementation, this type of behavior was never pondered.

3.3 Hospital

The Hospital's architecture for our simulations is different from what we had initially anticipated: instead of a hybrid architecture in which a user would directly be introducing new emergency occurrences, we ended up assuming the agent should rely on an automatic patient appearance on the system - without any need of external system's input until the end of the execution.

The agent's internal state is the environment's graph and the hospital's position in the graph. This information is especially important for the deliberative hospital simulation, as the agent uses this information to assess the correct distribution of the patients through the ambulance fleet.

Its actuators are simple: the agent can only dispatch emergency requests (assigning them to the ambulances), solve patient's conditions and charge the ambulances - we opted for this type of functionality for it made sense this agent could also assist its subordinate agents once they deliver the patients to the hospital.

3.4 Charging Stations

The charging stations are static agents, generated at random positions of the network at the beginning of the simulation.

The agent's internal state is its position and its availability, evaluated by the capacity of that charging station - an initialization parameter that constraints the number of ambulances that can charge in that station at the same time.

There is a single actuator for this agent: to charge the ambulance. Once the ambulance arrives at its chosen charging station, it automatically starts to charge the ambulance.

As for the election of a charging station, it is picked by a simple mechanism: once an ambulance wants to charge, that agent finds the closest and available charging station (has enough capacity for an extra ambulance). Once the ambulance chooses a charging station to charge in, its values are updated, more specifically, part of the capacity of that charging station is occupied.

3.5 Patient

As expected, the patient doesn't have any actuator, as its behavior is purely circumstantial - a patient's appearance is out of its understanding and from that moment forward only its internal attributes are used.

The agent has a single state: if it is already assigned to an ambulance or not, for internal verification purposes. Patients don't have any actuators.

3.6 System's Simulations

Our project utilizes each of the mentioned agents and applies to each of them a different set of logics, resulting in two distinct simulations: the Simulation *Hospital-Deliberative* and the Simulation *Ambulance-Collaborative With Reinforcement Learning*. The names are elusive to both's paradigms - they will be better described in the next subsections.

The number of patients (#P) generated during each simulation will be given by the following formula:

$$\#P = \#TIKs / Max\_Weight\_Network * 2.5$$

3.6.1 *Simulation Hospital-Deliberative*

The *Hospital-Deliberative* simulation's main concept is that *the hospital is responsible to assign a goal to each ambulance*. In this simulation, the hospital is a deliberative agent with access to the states of all agents, with enough autonomy not to require any input besides for the agent's information.

To improve the real-life scenario recreation of this simulation, the ambulances start off parked at the hospital. In addition, the

patient's appearance is immediately detected by the hospital - this implementation choice is simple: a patient in any emergency system is only taken into consideration once it has made an emergency request to the local hospital.

In each timestep, the hospital iterates through all unassigned patients and, if possible, assigns an ambulance to them. The only condition an ambulance has to fulfill in order to be assigned to a patient is to *be available* and *have enough energy to pick-up the patient and return to the hospital*. If such a condition is verified, the ambulance's goal is set to pick_up_patient (see Figure 2 of section 3.2).

In the same timestep, the hospital is also responsible to iterate through all ambulances to see if they need to charge. For each ambulance with energy below 50% capacity, its goal is set to go_to_energy.

Finally, whenever a patient is delivered to the hospital, the ambulance checks if its energy is below 50%. If so, it utilizes the hospital's charging station to recharge their battery.

3.6.2 *Simulation Ambulance-Collaborative With Reinforcement Learning*

It can be said several machine learning methods, in particular supervised learning algorithms, are not easily applied to real-life domains since they usually assume that agents are already provided by a predefined reward, which tends to a correct action by the agent for a given situation. We had modelled a reward-based approach, so the agent can estimate the optimal action-value function $Q^*(s, a)$. In order to do so, the agent maximizes a real-valued reward, accomplishing the mapping of situations to actions [4]. As an example, the goal of going to a position fairly close to a recent emergency case of a generated patient is given by its priority times ¾ for each unit of distance on the graph- as a result, the reward is decreased if the agent gets further, at least 0 of reward if the distance is greater than 25, we do not consider negative reward, penalty. The task of the ambulance is to perform a sequence of actions, observe their consequences, and learn a control policy, $\pi: S \rightarrow A$, in such a way that, from the initial state, it chooses actions that maximize the reward accumulated over time.

With that said, each ambulance influences the distribution of training examples by the action sequences it chooses, and therefore, the learner faces a tradeoff in choosing whether to favor exploration of unknown states and actions to gather new information, or exploitation of states and actions that it has already learned will yield high reward, to maximize its cumulative reward. And so, we came up initially with 2 solutions: ε-greedy and Soft-max. The first one chooses a random action with probability ε, so it can explore, and chooses the option which so far seems to be the best with probability 1-ε, so it can exploit. The parameter ε is decreased through time to allow higher exploration rate at the beginning of learning and more exploitation in the end. On the other hand, Soft-max varies the action probabilities as a graded function of the Q function, considering, in our case, the Boltzmann distribution. Its measure gives the probability that a system will be in a certain states as a function of state's energy and the temperature of the system:

$$\frac{e^{Q(s,a)/\tau}}{\sum_{b=1}^{n} e^{Q(s,b)/\tau}}$$

Making the analogy to our agents, the positive parameter temperature τ will be decreased throughout time to cause a greater difference in selection probability in favor of the best actions by the end of learning.

Nevertheless, we had implemented a third approach, selective, which gives an initialization to the ambulances to learn the localizations with higher probability of an emergency and with higher priority. During that phase, the ambulances have a full exploration behaviour, so they can apply exploitation after enough iterations, to the "real" patients given by the user (or as default)- in order to maximize the results of successful cases. Otherwise, if a phase of exploration wouldn't have been implemented, with a high number of patients, the ambulances would not have enough time to explore, and with a lower number of patients, the ambulances would not have many opportunities to learn by.

Interestingly, because the number of actions are equal to the number of states, since the possible actions are getting to a new position, a state, and being the unique actions to learn, we get two distinct properties:

1 - The Q function is a square matrix, described below

2 - The available actions do not need to be stated on a list, an integer is enough to know in advance which are the possible actions.

Hence, the action i, i>0, results in getting to the state i, wherein the value of i is between 0 and the number of the nodes of the graph minus 1.

Let us define the evaluation function Q(s, a) so that its value is the maximum discounted cumulative reward that can be achieved starting from state s and applying action a as the first actions. Thus, the optimal action in state s is given by

$$\pi^*(s) = argmax_a Q(s,a)$$

Concluding that learning the Q function corresponds to learning the optimal policy.

For our Reinforcement Learning we had applied to different methods to learn an *action-utility function*, *Q-Learning* and *SARSA*, stated thereunder. The user has the option to choose which method he intends to use, having Q-Learning as the default learning approach.

3.6.2.1 *Q-Learning*

Q-Learning method learns an action-utility representation, with no need to possess a model, either for learning or for action selection, hence model-free method. The update equation for Q-Learning is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

which is calculated whenever action a is executed in state s leading to state s.

3.6.2.2 *SARSA*

SARSA, for State-Action-Reward-State-Action, has the following update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

where a' is the action actually taken in state s', applied at the end of each s, a, r, s', a' quintuplet. Notwithstanding the fact that is quite similar to Q-Learning equation, whereas Q-Learning backs

up the best Q-value from the state reached in the observed transitions, SARSA waits until an action is actually taken and backs up the Q-value for that action. When exploration from the ambulances is happening, Q-learning pays no attention to the actual policy being followed, since it uses the best Q-value, thus an *off-policy* learning algorithm, while SARSA is an *on-policy* algorithm.

3.7 Display Menu

Executing the developed emergency response simulator, it will show an initial menu with a brief explanation of its functionalities, offering the possibility of using a randomly generated environment or a customized one.

Environment for the simulation:

1 - Use a randomly generated environment

2 - Generate environment

Triggering the customize option, the user will be asked to input the number of nodes and the minimum and maximum weights between two nodes.

After deciding which environment to use, the user will be taken to the main menu, where the type of simulation can be chosen. Each of these simulations were thoroughly explained in section 3.6.

Pick the type of simulation:

1 - Hospital assigns ambulances

2 - Ambulances cooperate with one another

9 - Change metrics

For each of the simulations, the user will be provided the option to use the default values for the *number of TIKs* (timesteps), *number of ambulances* and *number of charging stations* or customize them.

In the second simulation, the user will also be able to select between a set of policies and approaches to use in the reinforcement learning. There are three possible action selection policies - eGreedy, softMax or selective - and for the learning approach there are two - QLearning and SARSA.

This menu also offers the possibility of changing the metrics used in the simulation, which will redirect the user to a third menu.

Metrics - Choose your own metrics for the simulation

1 - Percentage of Each Priority in Patient Generation

2 - Time for Ambulance to Successfully Reaching Patient

3 - Number of Generated Patients During Exploration

Selecting the first option will enable the user to change the percentage associated with each of three priority types that will be used while populating the system. These priority types will be discussed later on.

For the second option, the user will be able to decide which is the specific time period for the ambulances to respond to it, depending on the patient's priority type.

Lastly, the third option enables the user to change the number of patients to be generated during the reinforcement learning's exploration phase.

3.8 Default Parameters

There is a set of values assigned for each intervening part in the simulations. They alter the characteristics of the environment and impose different restrictions on the agent's behavior. The parameter value inputs initialize all intervening system parties and create the necessary data structures for the system to execute by.

If the simulation does not receive any type of parameter specification, our systems' execution has a predefined set of single valued or interval parameters. Once the specialization of the parameters is selected in the menu, the value range for each parameter is asked as input and then used in the simulation's execution.

3.8.1 *Network*

The network has 3 three major parameters used in each simulation: the number of nodes of the environments and the minimum and maximum values for the weight of the edges.

Each of these values can be altered in the designated menu option [section 3.1]. If the user doesn't provide any input, the values for each of these parameters are represented in the table below.

Number of Nodes	15
Edge Minimum Weight	1
Edge Maximum Weight	5

Table 1. **Default values for the network generation**

The choice of such values came from the analysis of [1].

3.8.2 *Ambulances*

«In a one-tier system, an advanced life support (ALS) unit responds to and transports all patients who use 911 to activate the system. (...) One-tier systems average one ambulance per 53,291 population having an average response time of 7.0 minutes» [2].

Taking into consideration the values above and considering that, for simplicity purposes, we contemplated each node to contain an average number of citizens between 8000 and 10000, we defined the following formula to generate the number of ambulances depending on the number of nodes (N) of the network:

For $N < 15$, we generate a single ambulance. For a larger number of nodes, we generate a random value between *round(N/10)* and *round(N/5*).

To justify the fitness of the above formula, using the default value for the number of nodes of the graph N=15, we observed:

The number of ambulances (A) generated by our formula would average 2.5. The citizen number (C) for the fifteen nodes would average 135000. Dividing C per A we have an ambulance per 54000 citizens.

3.8.3 *Hospital*

The hospital is a single entity in the system. Its behavior is static and its responsibility varies with the type of simulation, as previously described.

Despite being singular, this agent has a random positioning on the network and, consequently, does not have any specific consideration of parameters to be specified.

This approach was deliberate despite the fact an intuitive assumption would prefer a centralized positioning as it could be more beneficial to the overall outcomes. This reasoning was overruled by another higher value: we opted for a random positioning technique for a more realistic environment sample.

The environment is a mere representation of a real-life scenario - it simply restricts the study to a specific part of a real life location. This restriction sometimes makes the hospital's location seem centralized or decentralized, when in fact, this way we can reflect on the emergency response's behavior as a whole for furthest locations

3.8.4 *Charging Stations*

We generate isolated charging stations depending on the number of ambulances (A), using the following formula:

For $A < 5$, we generate a single charging station. For a larger number of ambulances, we generate a random value between *round(A/5)* and *round(A/3)*.

Capacity wise, every charging station has a limit of 5 ambulances to accommodate.

The hospital has its own charging station. Its capacity is enough for any ambulance in the system.

3.8.5 *Patients*

Patients need to be randomly generated throughout the system's execution with a specific priority attribute - this attribute enables the computation of the amount of time an ambulance has to pick it up.

The randomization of its appearance is easy to compute: the system takes the total amount of time needed in a run, the desired frequency of patients' appearance (an input parameter) and generates random time periods for them to appear in (within the time limit indicated for that run).

As for the priority parameter, we had to reflect on the frequency of each priority case. We engaged in a probabilistic approach: if the user can specify the probability of the appearance of a patient with a certain priority. If the user does not indicate any frequency probability, the patient has a randomly assigned probability.

Lastly, the system assesses the time an ambulance has to pick up that patient. For this, we tried to find trustworthy documentation evaluating the required time for each type of emergency, concluding that:

In the United Kingdom's National Health Service, «The target of reaching 75 percent of patients in a life-threatening condition (Category A) in eight minutes is still in place (...). All non life-threatening calls to the Service will be put into one of four categories. Patients in the first two of these will still be attended by ambulance staff in either 20 or 30 minutes, depending on their condition. Those patients in the other two least serious categories could in the first instance be offered advice over the phone by trained ambulance staff» [3].

Consequently, we found that an appropriate form to compute this time would be to take into account the average distance between each location (average_weight) and the patient's priority:

If we call AW the average_weight between the nodes of the network and P the patient's priority type, we define the following minimum and maximum values for the emergency to be responded to, randomly selecting a time period between the two:

*randint(round(AW/P*5) , round(AW/P*10))*

To justify the fitness of the above formula, using the default values of the graph, having an average weight of AW = 3, we observed:

	Limit Time Interval
High Priority	[5 , 10]
Medium Priority	[7.5 , 15]
Low Priority	[15 , 30]

Table 2. Limit time interval for each priority with AW = 3

4. RESULTS

The evaluation follows the scientific method: since, in order to evaluate the simulation's performance to compare the different approaches to the emergency response problem, we execute each simulation changing a single parameter, fixing the others. This way we can demonstrate with certainty which are the most appropriate values and compare the simulation's fairly.

The notion of success used across the discussion of the results has the following logic: an emergency response is found successful if a patient is picked up before the priority time timestep assigned for that patient.

4.1 Number of Nodes

One of the simulation's key environment parameters is the number of nodes. Reflecting on its possible impact, we found that the modification of this value should not be significant to the average success rate. This holds, because despite the rise of the number of ambulances with the number of nodes, the increase of the average distance between the patient and the hospital mitigates the potential success increase.

In Figure 3 we present the plot results for the *Hospital-Deliberative* simulation. We increase the number of nodes in an arithmetic succession of a sum of 15 nodes. The number of ambulance's is proportional with the number of nodes (as described in section 3.8).

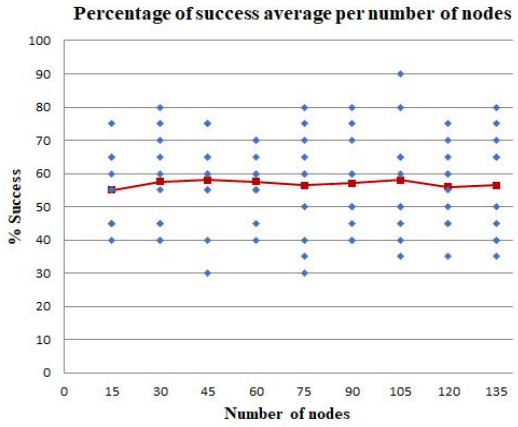

Figure 3: Plot representing the percentage of success varying the number of node

By analysing its variance across the multiple variations the following conclusions can be taken:

- Increasing the number of nodes for fixed parameters does not result in significant changes to the average

success rate of a simulation - the wine-red line tends to be invariant.

- Contemplating the results, the variation of the success rates for the same node number could be due to the graph disposition. The position of the hospital deeply influences the average distance an ambulance has to travel and, consequently, it impacts the time it has to pick up the patient, leading to higher or lower success values.

Despite the fact all of the conclusions are taken by the first simulation, these should also be similar to the second simulation, as its parameter variation describes the same proportions.

4.2 Number of Patients During Exploration

Another consideration of our study was focused on the second simulation. The number of patients generated during the exploration phase plays a major role in the success rate of the *Ambulance-Collaborative With Reinforcement Learning* simulation. This parameter rules how well the ambulance's learn where they should be positioned to have a bigger success rate.

To study how different numbers of patients impact the success percentages cases, we decided to set the number of TIKs to 500 and the number of nodes to 45. In Figure 4 we present the plot results for this study.

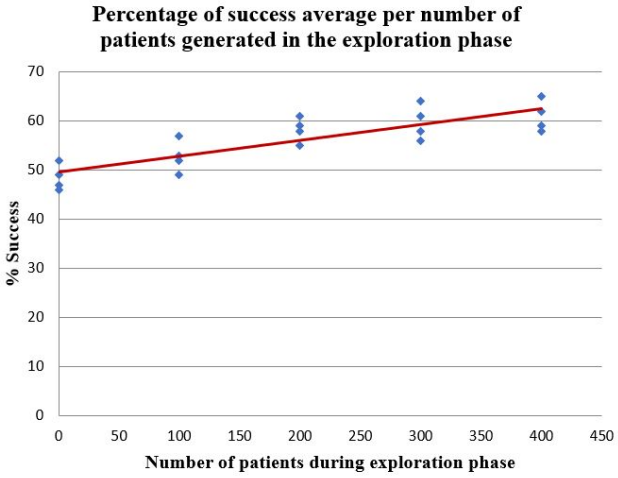

Figure 4: Plot representing the average percentage of success per number of patients generated in the exploration phase

By running the simulation with higher values for the number of patients, we detected a stabilization of the success rate of the system for the *Ambulance-Collaborative With Reinforcement Learning* simulation. This equilibrium is due to the fact the exploration phase reaches a plateau once it converges to a set of satisfactory learning positions for the ambulances.

4.3 SARSA vs QLearning

The *Ambulance-Collaborative With Reinforcement Learning* has two learning models at hand: Q-learning and SARSA. It was important for our evaluation to expose the best learning approach in order to harness all of the potential of this simulation.

Since the graph disposition varies for each simulation's execution, averaging the individual success rate of each learning mechanism is impractical. The only way to assess their performance is to compare them to each other for the same graph.

By varying the number of patients in this simulation and alternating between the Q-learning and SARSA learning (for the same graph) we can conclude the following:

- Q-learning's success rates present better values than the ones found on SARSA.
- The Q-learning values have higher variance between the success rates.

The plot depicts these conclusions. It illustrates the average success rate for 400 patients during the exploration phase, with 500 tiks, 100 nodes and the default simulation parameters. The red line illustrates the Q-learning results and the blue one relates to the SARSA conclusions.

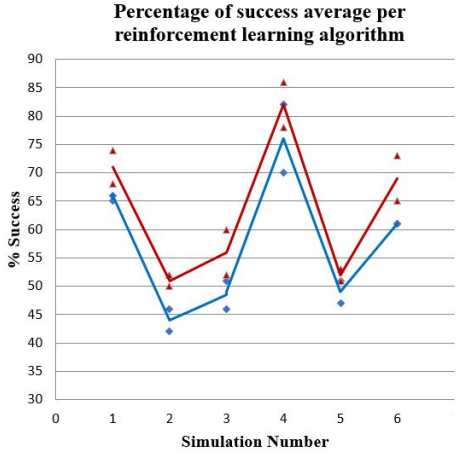

Figure 5: Plot representing the average percentage of success for the same graph for the Q-learning and SARSA learning algorithms

4.4 Selective VS eGreedy VS softMax

Next, for the *Ambulance-Collaborative With Reinforcement Learning* simulation, we had to experiment which was the best action selection policy.

We run our simulation for each policy (Selective, eGreedy and softMAX) with 45 nodes, 200 patients in the exploration phase and with the Q-learning learning mechanism (as this was the best learning mechanism we found and we aim to a full parameter optimization with these evaluations).

After executing the simulation for each of them, we concluded the variations between the three weren't significant enough for them to be ranked comparatively - they are all equally fit to aid the emergency request system.

4.5 Simulation Comparison

To compare the two distinct simulations we used the SARSA learning mechanism for the second simulation. Picking the worst learning mechanism (see section 4.3) would serve to prove that even with the worse performant one, this simulation still out bettered the first. As for the action selection policy we used the Selective one - since the action selection policy wasn't relevant to the second simulation's performance, this could be any policy.

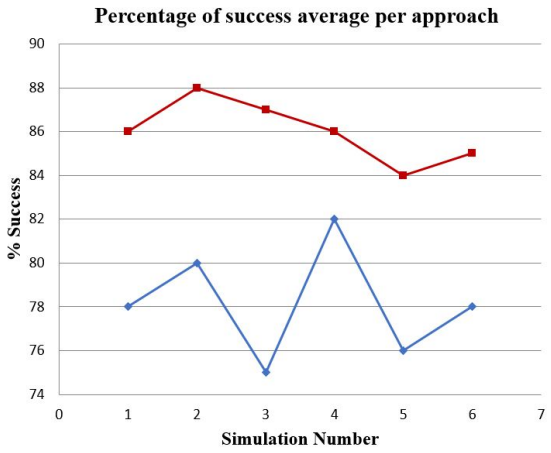

Figure 6: Plot representing the average success percentage per simulation

After several simulations, we were able to conclude that the learning approach - *Ambulance-Collaborative With Reinforcement Learning* - presented better values than the *Hospital-Deliberative* one. This difference wasn't as meaningful as we were expecting, but we can account for this outcome because of the difficulties of the system to isolate the parameters.

The high variance between the values of the first simulation prove this is an unstable performant approach, another reason to go for the steadiest simulation.

5. CONCLUSIONS

By assessing the results of our experimental simulations, we can deduct a series of conclusions.

Firstly, by assessing the parameters that have the biggest impact on our agent's success percentage, we can conclude:

- The number of node variations isn't relevant to the system's performance, since the agent's abundance is proporcional.
- The number of patients in the exploration phase converges for values bigger than 500 patients.

Secondly, this study realizes the understanding that a multi-agent system recurring to learning mechanisms on top of a reactive structure, has better results than a purely reactive one. Additionally, the simulation *Ambulance-Cooperative With Reinforcement Learning*'s solution, despite overtaking the *Hospital-Deliberative* simulation's results, has a tradeoff, between the agent's time period to explore and its successful patient recovery - it only offers higher success rates once its "learning phase" is concluded.

Lastly, by analysing each graph generated during the making of parameter studies and comparing it to its success rate, we can conclude how important a central hospital positioning in the graph is to the emergency response success rate.

5.1 Future Work

Following our system's conclusions, we can enumerate a series of improvements on the work we have developed for this project. These improvements would be incremental and its objectives would be either to introduce more features to the system or to extend the evaluation's performance analysis.

Firstly, running the various simulations with different pairing parameter variations would be interesting to test the dependence and interconnection between each other.

An implementation of negotiation between the ambulances would introduce the possibility to test if such a strategy could benefit an emergency system. The only restriction in such simulations would be not to harm the patient's responses with too much communication between each other.

Considering the introduction of different types of ambulances for different sorts of responses could potentially offer better performance for our system.

We also found that adding acceleration to our ambulances could be a curious approach. The velocity of the ambulance would be a tradeoff between the ambulance's energy consumption.

Another aspect to make notice is how the system's outcomes could be more conclusive. The system had some troubles during the evaluation phase due to lack of parameters stabilization. A more careful control of these values could produce more reliable conclusions.

Finally, we could also perform a proper input sanitization to avoid execution crashes if the user's input doesn't match the expected values. As a general public's usage was not considered in this project, we didn't find it necessary to have this sort of concern. Nevertheless, this aspect is critical for unguided usage of our system, once it is brought to a wider audience.

6. REFERENCES

[1] The structure of interurban traffic: A weighted network analysis. Consulted through ResearchGate's publications.

[2] Characteristics of Midsized Urban EMS Systems. Consulted through USA's National Library of Medicine webpage.

[3] London Ambulance Service. Consulted through NHS.

[4] Wooldridge, Michael. 2007. *An Introduction to MultiAgent Systems* (2nd ed.). Chichester, West Sussex.

[5] NetworkX: Python software for complex networks.

[6] Russel, Norvig. 2010. *Artificial Intelligence A Modern Approach* (3rd ed.). Upper Saddle River, New Jersey.