Emergency responses
AASMA Project Report Group 13

DUARTE SANTOS, MIGUEL GONÇALVES, and SOFIA CARVALHO, Instituto Superior Técnico

In this project we will be addressing the problem that emerges when trying to respond to various emergencies in a city, focusing on how to coordinate the available emergency services in order to do this in the most efficient way possible. We will strive to model this issue using a multi-agent system where each emergency service corresponds to an agent. These agents will operate in a simplified version of a city, portrayed by a squared grid where emergencies may appear randomly. They must cooperate to quickly and efficiently attend to these emergencies. By modeling this issue we expect to come up with a service that is able to solve it automatically.

ACM Reference Format:
Duarte Santos, Miguel Gonçalves, and Sofia Carvalho. 2021. Emergency responses AASMA Project Report Group 13. In ,. ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION AND PROBLEM DEFINITION

Every city, region or country faces the problem of efficiently managing its emergency resources and dispatchers (police stations, hospitals, etc.). A government or administration will want to assure that every emergency is responded to and that it is done in the most efficient way possible. However, this management is not trivial.

The relevance of this project comes, in fact, from the importance of a good management of emergency responses when administering a city. Emergencies are a sensitive matter as they are by definition a "call for help", thus failing to respond to one is an alarming issue. Therefore, it is in a city's best interest to have a good management of its resources to be able to reliably respond to emergencies quickly and efficiently. A tool that could calculate this would provide a more efficient management plan as well as assure fewer failures, making it very helpful for a city. Furthermore, it could eventually be used to identify resources that are in excess - helping the economy by keeping the money spent on resources to the minimum necessary.

We are interested in exploring the capability of a multi agent system of cooperating and working in a coordinated way to achieve a common goal efficiently. As such, we developed a multi agent system that attempts to solve the problem described earlier. Our project's main objective is to develop a system that attends to as many emergencies as possible while minimizing the time needed to do so. Therefore, we developed our agents to find the optimal way of managing their resources when responding to a sequence of emergency calls.

We mainly focused on emergency overflows - that is, when an agent's resources are not enough to deal with every emergency that is being assigned to it. In those corner cases, we want our agents to be able to prioritize emergencies that may be more urgent or more critical than others. Additionally, we want them to be able to decide to ask for help from other agents that may have the needed resources.

,

2 CONCEPTUAL MODEL OF THE SYSTEM

2.1 Introduction

This multiagent system attempts to replicate different emergency dispatch services in a city and explores different behaviours and methods of responding to emergencies efficiently. There are three types of agents, representing emergency dispatch services:

- Fire stations, which respond to fire emergencies;
- Hospitals, which respond to medical emergencies;
- Police stations, which respond to police emergencies.

Each agent is in a fixed location of the world and has available a fixed number of units of their type, which are used to respond to emergencies. All agents share the same architecture, design and implementation.

2.2 Environment

The environment is represented by a squared grid, of 10 by 10 blocks, which houses two agents of each type in fairly opposing sides, so that each can be responsible for approximately half the grid regarding emergencies of that type. One agent of each type has a total of 8 response units at its disposal, while the second agent of each type has only 6. The initial state of the environment is depicted in Figure 1.

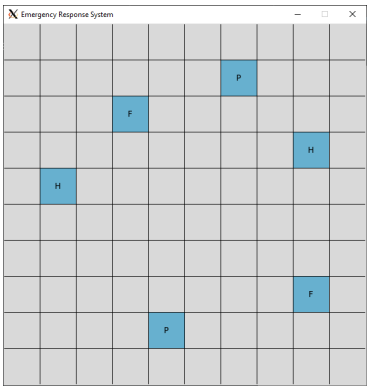

Fig. 1. Initial state of the environment, where 'F', 'H' and 'P' represent a fire station, a hospital and a police station, respectively.

Every second, zero, one or two emergencies are spawned with different probability (40%, 40% and 20%, respectively) in random free blocks of the grid. A spawned emergency has a 50% probability of encompassing each basic type individually (fire, medical and police), which means it has equal probability of being any combination of the three. It will also have one of four increasing severity levels, 1 to 4, with different probability (30%, 30%, 25% and 15%, respectively).

An emergency is considered to be answered when the needed number of units of each type arrives at its location. This number is defined by the severity level of the emergency and the basic types it encompasses. For example, a level 3 fire and medical emergency requires exactly 3 fire units and 3 medical units to be resolved.

Emergencies expire after being active and unattended for a certain period of time, which is defined based on the emergency types and severity level. While a higher number of encompassed types impacts this time limit positively, a higher severity level has the contrary effect. In other

words, the more severe an emergency is, the lower the time limit will be; but the more complex an emergency is (complexity as the number of types present), the higher the time limit will be.

As stated in the project proposal, and given everything described above, this environment is non-episodic, dynamic and discrete. It is also deterministic, as actions have no uncertainty associated, and partially observable, because agents have limited knowledge about active emergencies outside their area. In order for the environment to be truly dynamic we run each agent on a separate thread, so that the environment may evolve at its own pace independently of whether an agent is deliberating.

2.3 Agents

As mentioned before, agents represent emergency dispatch services and as such their goal is to respond to emergencies that appear around them. In order to identify these emergencies, all agents have a sensor which detects incoming help requests from emergencies of their type in their area (defined by closest proximity, meaning all blocks to which an agent is the closest one of its type). From this sensor, the agent also receives or deducts information about the state of a given emergency (active, answered, expired). A second sensor allows the agents to accurately locate their units in the environment.

Each agent has a fixed number of units of its type, which it uses to respond to emergencies detected in its area. Hence, all agents possess an actuator to send a specified number of units to a specified location in the grid, as well as one to recall its units back to the agent.

A unit can be considered an extremely simple reactive agent which, given a goal position, moves towards it. It has a sensor which determines its own location in the grid and an actuator which moves the unit one block in a specified direction. Despite this, every mention of agents in this report refers exclusively to emergency dispatch services and not to units.

Given these sensors and actuators, we experimented with different decision making behaviours to try to come up with a way for the agents to do their job (responding to emergencies) in the most efficient way possible. We implemented two different agent architectures, as well as an additional variant, which are further described next.

2.4 Agent architectures

2.4.1 Reactive agent.

We started by implementing a simple reactive agent architecture in which the agent follows the cycle:

(1) Perception: detects the state of emergencies in its area;
(2) Decision making: decides on a single action to perform (recall or dispatch units);
(3) Actuation: performs the desired action.

The agent recalls all units assigned to an emergency when said emergency expires and dispatches the necessary units to an emergency when a new emergency is detected. Both expired emergencies and detected emergencies work in queues, meaning emergencies are handled in order of detection in both situations.

This agent is designed to prioritize recalling units over dispatching them in order to optimize the use of its units and avoid having units unnecessarily unavailable. Finally, when dispatching units to an emergency, if the agent does not have enough available units to fulfil the needed amount, it sends the ones available and keeps the emergency in the front of the queue so it can send the rest when more units become available.

In summary, this reactive agent architecture is designed to recall units from the oldest expired emergency, or in case there are no expired emergencies, to dispatch the maximum number of units it can to the oldest active emergency (but never more than needed). A general diagram of this architecture is depicted in Figure 2.

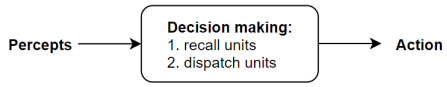

Fig. 2. Diagram of our reactive agent architecture.

2.4.2 Deliberative agent.

Having implemented a simple decision making behaviour, as described above, we then moved on to a more sophisticated agent architecture: the deliberative agent. We followed the BDI model, where the agent has a set of beliefs about the world around him, which include its position in it, the position of its units and the state of emergencies in its area. The agent constantly updates its beliefs with new precepts from its sensors, which collect information on new emergencies, the state of existing emergencies and its units' current location.

From its current beliefs, the agent determines a set of desired actions to perform, which will essentially be to recall its units from every expired emergency and to dispatch the needed number of units to every active emergency in its area. This set of desires is then filtered to determine which of these action the agent will in fact try to achieve, called intentions. This filtering is done based on the number of units available, the priority of active emergencies (defined by the severity level and the time remaining until expiry) and whether or not the agent would reach the emergency in time. We should note that, contrary to the reactive architecture, this deliberative agent only selects intentions it can fully answer right away, i.e. send the entirety of the needed units instead of only a fraction.

Having selected its intentions, the agent then makes a plan on how to achieve them. It maintains the same action priority as the one used in the reactive architecture, for the same aforementioned reasons, resulting in recalling actions (in order of maturity) being ahead of dispatching actions (in order of priority) on the plan. The agent then consecutively selects the next action on the plan and executes it.

This architecture is significantly more sophisticated than the reactive one in the sense that as the agent prioritizes emergencies based on their severity level and time remaining, and does not waste resources on impossible emergencies or emergencies which he does not have the capacity to respond to at a certain time. However, this added computation makes this architecture potentially slower than the reactive architecture. It also runs the risk of potentially pursuing intentions which are no longer justified (like dispatching units to an already expired emergency) or constantly keep changing its intentions, which would be detrimental to its performance. To avoid these extreme scenarios and to try to make up for the computational overhead of deliberation (choosing desires, filtering them into intentions and making a plan to achieve them), these particular subtasks are not constantly computed. Instead, after deliberating once, the agent follows its plan until at least one of the following conditions happens:

- new units have become available and there are active emergencies;
- an active emergency has expired;

- the current plan was been fully executed.

Only when one of the previous situations are verified does the agents deliberate, allowing it to act faster but still be flexible enough to changes in the environment. A general diagram of this architecture is depicted in Figure 3.

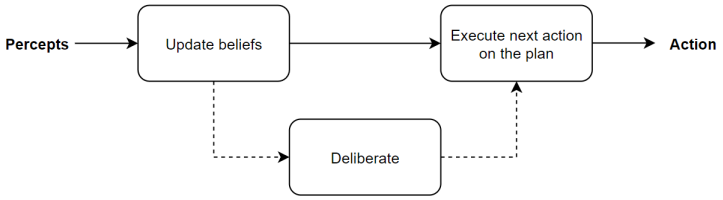

Fig. 3. Diagram of our deliberative agent architecture.

2.4.3 Social Deliberative agent.

The last decision making behaviour we implemented is a variant of the deliberative agent architecture, this time with social agents. By adding a sensor to receive help requests from other agents as well as an actuator to perform said requests, the agents are now able to communicate, allowing them to cooperate in responding to emergencies in each other's areas.

This new deliberative agent architecture works exactly the same way as before, with the addition that the agents' beliefs now include the existence of other agents and their positions in the environment. This way, when asking for help, an agent asks only the second closest agent to the emergency (itself being the closest, hence the emergency being in its area). This is helpful when there are more than two agents of the type in question (which in truth does not happen in our simulation).

The deliberation process is essentially the same, with an additional action: ask for help, which sends a message to a specific agent with knowledge about an emergency outside their area. The agent is designed to only ask for help when it has some available units but not enough to respond to its next priority emergency. Therefor, the intention of asking for help actually emerges from the desire of dispatching units, and this type of intention will appear at most once in a given plan.

When updating its beliefs, the agent now also uses information from its new sensor, which receives help requests with information about new emergencies. These are then treated in the deliberation process the same way as detected emergencies.

2.5 Graphical interface

In the early stages of our project, we developed a Graphical User Interface (GUI) that displayed a grid - representing the world - as well as all the dispatchers (agents), units and emergencies, present in the environment.

The existence of the GUI was extremely useful in development, especially while developing the different architectures. It allowed us to quickly and easily understand what was happening in the world, in each epoch. For example, it allowed to see what type of emergencies were popping up and where they were, and also what was the plan being executed by each agent, to combat these emergencies.

Despite this, it began miss-behaving when we decided to run each agent in it's separate thread. Ultimately, after many attempts in bringing it back, we decided to deprecate it, and because of that,

it is no longer possible to graphically display the execution of our project. Even so, a command-line representation of the world and it's contents is still used in production.

A screenshot of the GUI can be seen in Figure 1.

3 EXPERIMENTS AND RESULTS

3.1 Definition of experiments

In order to compare the different agent architectures and take more objective conclusions from our project, we decided to perform some experiments. Each experiment was performed 3 times, one for each of the agent systems that we developed.

The experiments we performed essentially consisted on first running the project several times and then observing the results obtained, comparing them for each architecture. To do so, we embedded some evaluation metrics in the code to help us track the performance of the agents on a given run. The metrics that were set are briefly described in the table below.

We consider a 'run' to be one execution of our program with an agent system, for a fixed number of time-steps. Between different runs, the environment will inevitably differ in the number and position of the spawned emergencies, which makes it possible to test the agents' decisions in different situations.

Specifically, we decided to perform 50 runs for each multi agent system, where each run went on for 100 time-steps. In each run, there are 2 dispatchers of each type, where one has 6 units and the other has 8 units. In each step, 0, 1 or 2 emergencies are spawned with probability, severity and expiring time as described earlier.

We mainly focused on the *percentage of expired emergencies* and on the *mean response time*. We believe that they brought the most relevant information, as the main goal of the multi agent system, as mentioned before, is to respond to as many emergency calls and as quickly as possible. We also took special notice on the severity-specific versions of these metrics (for example, the *percentage of expired emergencies that had severity 4*). This is because it is important for our agents to correctly prioritize the most crucial emergencies. Finally, the *amount of calls for help* (of the social deliberative agents) is also very interesting, as it allows us to perceive how many times the agents were assigned an emergency that they could not answer and, thanks to their social ability, were able to transfer it to another agent.

	Evaluation Metric	Description
1	% expired emergencies	percentage of emergency calls not answered on time
2	...of which had severity level 1	% of expired emergencies that were of severity 1 (*)
3	...of which had severity level 2	% of expired emergencies that were of severity 2 (*)
4	...of which had severity level 3	% of expired emergencies that were of severity 3 (*)
5	...of which had severity level 4	% of expired emergencies that were of severity 4 (*)
6	% expired sev. 1 emergencies	% of emergency calls of severity 1 not answered on time (*)
7	% expired sev. 2 emergencies	% of emergency calls of severity 2 not answered on time (*)
8	% expired sev. 3 emergencies	% of emergency calls of severity 3 not answered on time (*)
9	% expired sev. 4 emergencies	% of emergency calls of severity 4 not answered on time (*)

	Evaluation Metric	Description
10	Mean response time	mean time between an emergency spawn and arrival of help
11	...to sev. 1 emergencies	mean time between a sev.1 emergency spawn and arrival of help
12	...to sev. 2 emergencies	mean time between a sev.2 emergency spawn and arrival of help
13	...to sev. 3 emergencies	mean time between a sev.3 emergency spawn and arrival of help
14	...to sev. 4 emergencies	mean time between a sev.4 emergency spawn and arrival of help
15	Amount of calls for help	# of times the agents asked for the help of another agent

** Lines 2-5 refer to the amount of the expired emergencies that had a given severity level (# expired level X / # total expired) while lines 6-9 refer to the amount of emergencies of a given severity level that expired (# expired level X / # total level X). Although being very similar, we found both these metrics to be relevant and, as such, they were both included.*

3.2 Hypothesis

In this section we will describe what we are expecting the experiment results to be, based on what we studied of the developed architectures and on the nature of our project's environment.

We believe that, overall, the social deliberative system will perform better than the deliberative system, which in turn will perform better than the reactive system. This is because the deliberative systems weight the possible actions that they can take, using the knowledge that they have on each emergency to prioritize their actions. Furthermore, the social system is thought to perform better as the agents have the ability to help each other when struggling.

- **Mean response time:** Should be lower in the deliberative systems, because the reactive system may waste a lot of time sending units to emergencies that will not be reached in time. Furthermore, it may be lower in the social system, because, when cooperating, the agents do not have to wait until they have enough units to respond to an emergency - they can simply ask for help from another agent. However, it may as well be higher because the agent that is helping can take longer to respond to that same emergency (eg. because it is further away or has more crucial calls to attend to first).
- **% expired emergencies:** Should be lower in deliberative agents as they prioritize emergencies that are closer to expiring, preventing that outcome. Should also be lower in the social deliberative system, as the agents can ask for help instead of letting an emergency that is impossible to answer expire.
- **Severity-specific metrics:** Optimally, these metrics should be lower for emergencies of higher severity (and may consequently be higher in less crucial emergencies). This should happen in deliberative systems - they are able to prioritize emergencies based on their severity, instead of a reactive agent that prioritizes them based on the time at which they are assigned to it. Should not change significantly between the deliberative systems because their policy in regards to the emergencies' severity is the same.

3.3 Results

In this section we present the results of our experiments, conducted as described in the previous section. This data was extracted from 50 runs of 100 steps, for each agent architecture.

Table 1 contains the mean response time (line 10 of the evaluation metrics table), as well as the percentage of expired emergencies (line 1), per agent architecture. It also contains the mean amount of calls for help (line 15) performed by the Social Deliberative agent.

Figure 4 and Figure 5 present their data in bar charts, in which each x-axis value is associated with three bars, one bar for each agent architecture. They analyse the mean response time per severity level (lines 11-14), and the percentage of expired emergencies per severity level (lines 6-9), respectively.

Table 2 contains the percentage of expired emergencies of a given severity level (lines 2-5).

Metric	Reactive	Deliberative	Social Deliberative
Mean response time	9.5144s	8.7410s	8.2620s
% expired emergencies	6.7980%	10.1678%	8.4206%
# calls for help	-	-	6.0667

Table 1. Mean response time and % of expired emergencies per agent system, for 50 runs of 100 steps.

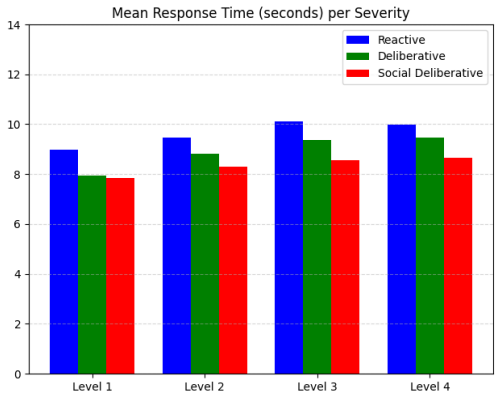

Fig. 4. Mean response time per severity and per agent system, for 50 runs of 100 steps.

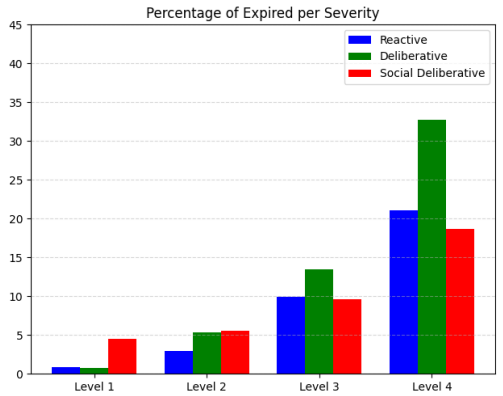

Fig. 5. % of emergencies of each severity that expired, per agent system, for 50 runs of 100 steps.

Metric	Reactive	Deliberative	Social Delib.
% of expired emergencies that had sev. 1	2.7570%	13.4892%	15.5280%
% of expired emergencies that had sev. 2	10.7648%	18.9758%	17.2884%
% of expired emergencies that had sev. 3	32.1498%	24.4520%	28.3072%
% of expired emergencies that had sev. 4	52.3282%	43.0816%	36.8752%

Table 2. % of expired emergencies per severity and per agent system.

4 CONCLUSIONS

In a general manner, the results of the experiments corroborate our aforementioned hypothesis about the performance of the different systems. We notice, however, a few exceptions.

In Table 1, the mean response time decreases throughout the different systems, as predicted. The percentage of expired emergencies, however, is surprisingly lower for the reactive agent. This is probably due to the simplicity and reduced size of our simulation. It is also possible that the deliberative systems spend too much time deliberating, leaving less time to attend to emergencies. The social deliberative system did improve on this metric when compared to the deliberative system, which suggests a clear advantage of cooperation in this multi agent scenario.

Figures 4 and 5 break down the previous metrics by severity, allowing us to better understand the different decision making behaviours. In Figure 2, the three MRTs (Mean Response Time) for all severity levels are similar to the overall MRT, but in Figure 3 this is not the case. In Figure 3 we are able to see a strong decline in the percentage of expired emergencies of higher severity levels, when comparing the Social Deliberative agent system with the other ones, as expected. On the contrary, the Deliberative system, that is not social, exhibits the worse performance out of the three architectures, which we did not predict. This is mentioned below.

From Table 2 we can conclude that, despite the overall percentage of expired emergencies being higher in the deliberative systems than in the reactive one, the fraction of these expired emergencies which are of higher severity levels (3 and 4) is lower, as in these systems emergencies are being prioritized mostly based on this parameter. This proves that the prioritization of emergencies by the deliberative systems has the desired effect. On top of this, having cooperation in the social deliberative system allowed for an additional decrease in the severity level 4 portion of expired emergencies, as agents work together to more efficiently respond to these top priority emergencies.

In conclusion, the deliberative multi agent system with social capability was the one that performed better overall. However, we realised that a deliberative system without social capability did not perform that well comparing to the reactive agents. This is because a deliberative agent always prioritizes higher severity and more urgent emergencies - without social capability, this type of system will frequently choose to wait until it has enough resources to answer the higher priority emergency and avoid responding to emergencies that require many resources, leaving them unattended. This often results in a higher amount of expired higher level emergencies (which may even include the one that it was waiting for). In contrast, the socially capable one can simply ask the other agents to send their resources as well, and thus does not need to wait as frequently, while still correctly prioritizing the emergencies.