Multi-Agent Surveillance

Group 08
Yannick Damoiseaux, Roman Ilic, Johann Lottermoser, Joaquin Monedero Simon, Marie Picquet, Giacomo Terragni
Department of Advanced Computing Sciences
Maastricht University
Maastricht, The Netherlands

***Abstract*—This paper will discuss the implementation of a multi-agent surveillance system on a set of maps, using indirect communication. Furthermore, this paper will investigate the performance of agents in exploration and coverage, whilst having no previous knowledge of the map and communicating indirectly through the use of sounds and markers. Firstly, a priority-based task decider was implemented for both the guards and the intruders. Said decider is composed of a list of tasks which are paired with a priority value, and a decision is made based on the current state and what is perceived. Moreover, another intruder agent was created using Deep Reinforcement Learning, in order to improve the evasion from guards. The features used to exploit indirect communication are a vision memory, sound matching using a classification neural network together with an algorithmic approach, as well as pheromone markers. Finally, experiments were implemented for exploration and surveillance, in order to find the perception parameters that would maximize win percentage, as well as the effect of modifying the number of guards and intruders in indirect communication.**

***Index Terms*—Multi-agent surveillance, Pursuit- evasion game, Indirect communication, Stigmergic communication, A-star, Task-oriented agent.**

I. Introduction

In the last few years, automated surveillance has gained an increase in demands due to the fear amongst the population and a call for safety resulting from the growth of criminal and terroristic developments (Pramanik et al., 2017). In order to improve the security and safety of the individual, the high performance and efficiency of the surveillance is crucial. In this paper, an investigation on how to efficiently solve an automated multi-agent surveillance on different environments will be proposed.

A multi-agent surveillance is the exploration and coverage of an environment, i.e., a map, by different agents. In this paper, exploration refers to the discovery of the environment by the agents and coverage refers to the maximization of the area that is perceived by the agents in one time step (discrete time and space are assumed). There are 2 types of agents: the guards, which are the pursuing agents, and the intruders, which are the evading agents. Both agent types have no prior knowledge of the environment and can only communicate indirectly with the use of markers and sounds. The goal of the intruders is to reach a target area in the environment, whereas the goal of the guards is to arrest the intruders before they reach their target. The game ends when either all the intruders are caught or when an intruder is in the target area for more than 3 seconds.

In the purpose of the research problem, the following questions were discussed:

- What are the effects of using pheromones on the exploration and therefore coverage of the map?
- How does indirect communication affect the results of the surveillance game? Specifically how does the adding of sound affect these results?
- How does the number of agents affect the results of the simulation?
- How does a classification approach compare to an algorithmic approach when detecting sounds?
- How does a Deep Reinforcement Learning evasion approach compare to an algorithmic approach?

Due to its importance, a broad range of research has already been conducted on multi-agent surveillance systems. Algorithms in the domain of Machine learning such as an Artificial Neural Networks have already been implemented. Artificial Neural Network for surveillance systems can mimic the behavior of living beings such as ants (Nguyen, 2021) and fish (Rossi et al., 2018). This is especially interesting for the intelligent cooperation of the agents in order to decentralise multi-agent learning.

Other research shows how different algorithmic approaches such as the A-star algorithm (Hart et al., 1968) or the leader-based approach using only non-communicative behaviour (Quinn, 2001) have given promising results. The leader-based approach follows the idea that if an agent sees another agent of the same type, it can foresee the action of the other agent, as they have the same ”brain”. The agent can then take consequent actions and a leader is chosen depending on their location.

Both agent's algorithms were implemented by combining different algorithmic approaches in priority-based task implementation. The agent will follow different algorithmic approaches based on the given priority of the task. Moreover, a second approach was implemented for the intruder agent, namely Deep Reinforcement Learning.

The paper is organized as follows. In section II, the different methods used in this research are given and an explanation on the implementation of the different agents as well as the different features is provided. This section will present

the intruder and guard algorithms consisting of the priority-based task implementation using different goals as well as another the Deep Reinforcement Learning approach for the second intruder agent. This section will furthermore introduce, the multiple features that were implemented to improve the performance of the agents such as markers and sounds. Section III is dedicated to the experiments performed with the different agents and parameters and section IV will present the results of these experiments. Finally, the results will be discussed in section V and the conclusion will be presented in section VI.

II. Methods and Implementations

This section introduces the methods and corresponding implementations that were used in order to approach the problem statement. It should be mentioned that for all pathfinding problems we use the well known A^* algorithm by (Hart et al., 1968). First, the general features and functionalities will be described. After that, a deep dive will be taken into the methods and their implementations proposed for both the guards and the intruders.

A. Features

In order to make an interesting simulation to test our guards and intruders in, several special features were implemented next to standard features such as walking and rotating.

1) Graph representation: A graph is used to represent the knowledge the agent has on the map. Tiles are represented by nodes, whilst edges represent a walkable path. The agent creates nodes and edges when unexplored tiles are in vision. The graph is stored as Hash map for accessibility.

2) Teleports: Teleports can be used by guards and intruders. Whenever stepped on, the agent is taken to a new tile within the map. They are unidirectional, meaning that they cannot be taken to go back. The teleport is also added to the graph, but only an edge from the entrance to exit is stored to represent the unidirectional property.

3) Agent vision: To calculate the visible field of an agent the surrounding area, a vision angle α and a maximum viewing distance γ are needed as input. This results in a cone shaped area in front of the agent which is the already mentioned visible field. For calculation ray-casting is used. A ray is a line which originates at the agent's position and is being cast for a certain distance in the direction in which the agent is looking. If the ray was to hit an object that is vision blocking (e.g. a wall) it is stopped. Everything that is then touched by the ray will be visible. In ray-casting many rays are sent out to cover the whole area that should be visible. Since we assume discrete space, a line is a collection of points. To calculate those points, we use linear interpolation between the agents position and an endpoint which is γ away from the agent. When the vision angle $\alpha = 90$ then the agent will be able to see $\alpha/2$ in both directions from the direction the agent is currently looking at.

Algorithm 1 RayCasting

Input: vision angle α, max vision distance γ, agent direction β, $surroundingArea$

Output: the visible field of the agent

```
endpoints = calculateEndpoints(α, γ, β)
for all endpoints do
    ray = castRay(endpoint)
    if ray is interrupted in surroundingArea then
        stop ray and set everything currently on the ray to visible
    else
        set everything on ray to visible
    end if
end for
return everything that was set to visible
```

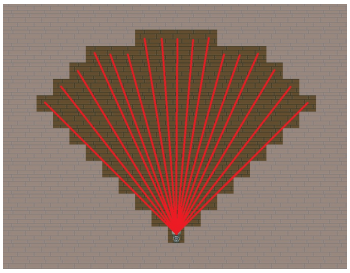

Fig. 1. Example of rays being cast from an agent

Not only walls influence the rays but also shaded areas. A shaded area reduces the vision in that part of the map. Once a ray hits a shaded area tile, the distance the ray will travel is reduced. This means that when standing outside of a shaded area the agent is only allowed to see limited distance into the area and when being inside the area the viewing distance will be reduced from the beginning.

4) Sounds: Sounds add an extra dimension to recognizing and detecting agents of the opposite team. A sound is automatically generated by the game controller when necessary. In the following cases, also referring to as sound types, a sound is generated:

- When an agent walks
- When an agent rotates
- When a guard sees an intruder
- When a guard catches an intruder

The last two sounds are also known as guard yells. These guard yells can only be understood by the guards. In other words, the intruders do not anticipate on this sound. Each sound type has its own distance it can be heard from, where the two guard yells always have the same maximum distance they can be observed from.

Agents receive sounds in the form of $sound = \{\theta \in [0..360), \zeta \in [0, 1]\}$, where θ is the angle in degrees and ζ is the loudness of the sound. Note that the greater ζ, the louder the sound, the closer the sound source is. To prevent agents knowing the exact angle θ, a normal distributed uncertainty is added with $\sigma = 10$. For computing the loudness ζ, the distance between the agent and the sound source needs to be calculated. Since we assume agents cannot hear through walls,

two different cases have to be considered when calculating this distance. The following pseudocode points out these cases.

Algorithm 2 SoundDistance
Input: both the positions of the agent and the sound origin
Output: the distance from the agent to the sound
getDistanceToSound(PA, PS) is
1: **if** (wall in between PA and PS) **then**
2: **return** length of A^* path between PA and PS
3: **else**
4: **return** euclidean distance between PA and PS
5: **end if**

5) Pheromone markers: In our simulation we use pheromone markers as indirect communication between agents to improve their collaboration. Guards and intruders use distinguishable pheromones, which means that pheromone markers created by guards can only be perceived by guards and intruder markers only by intruders. Each agent will leave a trail of pheromones that can be perceived by other agents of the same type, if and only if the agent is close enough to the marker. Each pheromone has an associated strength that decreases with time according to the following equation: $strength[k+1] = strength[k] - (R \cdot T)$ where $strength \in [0, 1]$ and R stands for an arbitrarily chosen constant relative to the actual time-step T also called the pheromone strength reduction. The distance a pheromone marker can be perceived from is calculated with $distance = strength \cdot S$, where S is a constant called the initial smelling distance. The pheromone markers are mostly used as so called exploration markers (Nguyen, 2021), which means that agents will prefer going into a direction where there are the least pheromones. Next to this, we also use the direction of these markers to match sounds better. More on this will follow in later sections. This direction of the markers corresponds to a weighted average calculated with $\overline{\theta} = \frac{\sum_{k=1}^{m} \theta_n \cdot strength_n}{\sum_{k=1}^{m} strength_n}$ where m stands for the amount of pheromones being smelled.

6) Visual memory: In order to make our agents remember if, where and when they have seen other agents, we introduce the visual memory feature. Visual memory contains the following information for each agent it has seen: $visualmemory = \{\vec{p} \in \mathbb{Z}^2, t \in \mathbb{R}_0^+, o \in \{0, 90, 180, 270\}, at \in \{guard, intruder\}$, where $\vec{p}$ is the position where it has last seen the agent with the position being relative to the current position, t being the time in seconds how long ago it saw the agent, o being the orientation when it last saw the agent, at being the type of the agent it has seen. The visual memory is used in several deciding and task algorithms as you will read in the next sections.

B. Tasks

In order to make our agents able to use different versions of implemented algorithms, and allowing guards and intruders to share algorithms, we implemented so called tasks. Each task represents an algorithm that focuses on a subproblem an agent can encounter in our simulation. While tasks solve subproblems for our agents, we need a mechanism that switches between these tasks based on the observed state of an agent. This is done with the use of an alleged decider. To improve the process of task switching in the decider, each task has a fixed priority based on the subproblem it is created for. This means that tasks with a higher priority are preferred over tasks with a lower priority. Next to that, tasks can indicate whether they finished solving their subproblem.

First we will take a look at some general tasks and sound deciding that are used by both the guards and the intruders. After that, we will take a closer look at the guard and intruder specific tasks.

1) Frontier based exploration: Since the agents have no prior knowledge of the map they need an efficient way to explore the unknown space. In frontier based exploration the graph map representation of the agent is exploited. Frontiers are nodes which do not have four edges. If a node has four edges it is no longer considered a frontier since it has an edge in each direction and therefore all surrounding nodes are also explored. This is really helpful when one wants to discover the edges of the explored space to the unexplored space. To explore efficiently we always choose the closest frontier node to go to. To calculate the path to that frontier we are using A^* to get the shortest path possible in the current explored space. Since we do not store the walls in the agents graph, a self edge is added to prevent nodes being considered a frontier even though they are not anymore. In some cases it might not be possible to reach a node, for example when the agent is going through a teleport and ends up somewhere it has not yet explored. In that case we choose the next closest frontier to the agent and check if it is reachable by A^*.

2) Frontier based exploration in direction: In the surveillance game mode the intruder agents not only want to explore the map until they find the target area but they want to explore in the direction of the target area which is given to them. To achieve this with frontier based exploration we changed the heuristic by which the best frontier to go to is chosen. With the angle of the target area, the agent calculates an anticipated target area position, which is a certain distance away from its spawning point. Then the frontier which is closest to the anticipated position is chosen. Again we use A^* to compute the shortest path to that frontier. Once the anticipated target area is explored but the real target are is not yet known, a new anticipated goal is calculated with the same angle from the spawn point but a greater distance. This way we can ensure that the intruder will find the target area in an efficient way. In some cases it can happen that the anticipated area can not be explored since it is in a wall or maybe there is a part of the map which can not be explored because it is closed off. Because of this the agent updates the anticipated goal once a third of the surrounding area around the goal is explored.

3) Sound deciding: Sounds are important when it comes down to detecting agents of the opposite team. For a guard it could lead to finding an intruder, while for an intruder it could mean escaping fast enough from a guard. To make sure that agents only act on sound that is valuable to them, in other words, act on sounds that come from agents of the opposite

team, we created two methods that agents can use to decide if they should act on a sound or not. This procedure we also call sound matching, where a sound is matched if the algorithms thinks it belongs to one of the teammates.

- Algorithmic approach. To decide if an agent should act on a sound using the algorithmic approach we use the visual memory together with the pheromone marker angle (see section II-A5 on how to calculate this angle). For each teammate in the visual memory that we have seen more recent than a predefined number of seconds, which we defined as 2.5 seconds by early testing, we check if the difference between θ_{sound} and the angle where we have last seen our teammate is smaller than some constant, which we decided to be 50 while performing some early testing. When this difference angle is smaller than the defined constant the sound is said to be matched to this teammate. When there are multiple unmatched sounds we choose the sound that is the most in the opposite direction of the pheromone marker angle and is the loudest.
- Neural network. Next to the algorithmic approach we also created a classification neural network in the form of a fully connected feed forward network. The input is the sound angle, sound loudness, visual memory angle, visual memory orientation, visual memory time ago and visual memory distance. The output is true or false, where true means that the sound is matched to the visual memory of the input and false means that it is not. It should be observed that for each teammate the network is fed with the same data, except changing the visual memory. After feeding all visual memories to the network a sound is matched if there is at least one match found, otherwise the sound is unmatched. The neural network is trained with labeled data generated by our own simulation. At each step the input data needed, together with a label if the sound belongs to this visual memory, is created. The network consists of two invisible layers, where the first layer has five nodes and the second layer has four nodes. On the validation set the neural network achieves an accuracy of 84

C. Guards

In this section we will take a closer look at the tasks specifically created for the guards, along with the decider.

In this subsection we will look into the guard specific tasks that we implemented.

1) Pursuit: A guard will enter pursuit when it sees the intruder in the present. In other words, when $t = 0$ of the visual memory. We achieve a more multi-agent approach to pursuit by creating two different types of pursuit inspired by (Quinn, 2001). Namely, a close and a far pursuit. This approach relies on the fact that all guards are homogeneous and thus use the same policies. When a guard does not see any other guard while performing a pursuit it will always perform the close pursuit. However, when a guard sees that another guard is closer to the intruder he is pursuing it will switch to far pursuit. First we will look at the close pursuit.

In the close pursuit the guard will first try to move to a distance of x tiles from the intruder. After this, the guard will try to move x tiles in front of the intruder. x is calculated as $x = \frac{1}{3} \cdot \|\vec{p}_{intruder}\|$, where $\vec{p}_{intruder}$ is the position of the intruder relative to the current position of the guard.

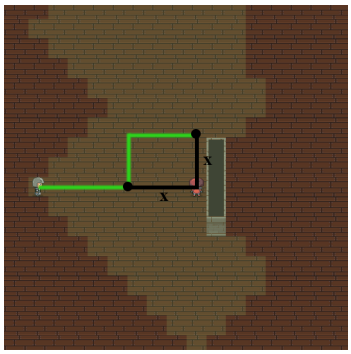

Fig. 2. An example of a close pursuit

Note that this x is a preferred distance, which means that it can still increase or decrease after calculating based on the reachability of the resulting tile. In figure II-C1 you can see an example of how the close pursuit could look like.

During the far pursuit the guard recognized that there is another guard closer to the intruder, so it will try to circumnavigate to cut the way of the intruder. The angle to the intruder and the angle to the guard, which is closer, are used to find a path. There are two scenarios:

- Other guard is to right
- Other guard is to left

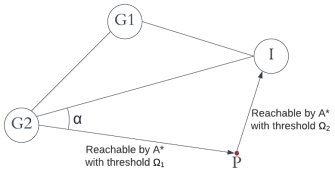

Fig. 3. Schema of a far pursuit, where $G1$ is in close pursuit and $G2$ is in far pursuit

The agent always wants to go to the opposite direction of the other guard. In Figure II-C1 this process is illustrated. $G2$ is doing $FarPursuit$ and will calculate a point with angle α such that he is approaching the intruder from the other side of $G1$. A criterion for choosing this point is if it is reachable from the position of the agent and if from that point the intruder is also reachable. This is done with A^* and a threshold on the length of the path. The threshold is there to ensure that the guard will not move to a different part of the map where the intruder might be reachable but not in an efficient manner. In case that these criterion are not fulfilled the angle α will be decreased and the process is started again. In the worst case the guard will choose the straight way to the agent because α decreased to 0.

2) Finding the origin of a guard yell: When a guard hears a guard yell it should try to find the origin of this yell, while it is guaranteed that there will be an intruder. That is why this task has the second highest priority after pursuit. Finding the origin of a sound comes down to trying to find a A* path, or performing exploration in direction, towards the direction of the sound. Due to the fact that exploration in direction only works if the sound originates from an unexplored area, first an A* path is tried to be found. To find this, first an upper and

lower bound are approximated by $upperbound = 50 \cdot (1 - \zeta)$ and $lowerbound = 25 \cdot (1 - \zeta)$. To not exactly know the distance, or to take into account changing distances we add a normal distributed uncertainty to both bounds with with $\sigma = (1 - \zeta) \cdot 10$. Note that the louder the sound, the smaller σ becomes. In the direction of the sound and between this upper and lower bound a tile is trying to be found that is reachable with A*. If there is a path found, the guard will follow this path. Otherwise, exploration in direction is performed into the direction of the sound.

3) Visiting the last seen positions of the intruders: After some time a guard will have explored the whole map. In the case that there are no visual or sound inputs to act on, the guard will perform this task. The internal working is just what the title of the task suggests. It visits, using A*, the positions where it has last seen each intruder. Note that intruders that are caught will still be in the visual memory.

D. Intruders

In this section intruder specific tasks are explained with focusing also on when those tasks are performed.

1) Evasion: Evasion is one of the most important tasks of an intruder. In this task the agent tries to escape something that it considers a threat. For the intruder two things are considered a threat:

- Seeing a guard at that moment in time.
- Hearing a sound that can not be matched to a team mate as explained in Section II-B3 and that has a loudness above a certain threshold.

Two different approaches were implemented:

- Algorithmic approach: The agent always wants to evade in the opposite direction of the direction the threat is coming from. Sometimes that might not be possible because of a wall, so the agent has to find the best direction to go to.

Algorithm 3 FindGoal
Input: angle of the threat α, map of the agent ($graph$)
Output: the best goal to go to

```
escapeAngle = getOppositeDirection(α)
possibleGoals = []
for i from 0 to threshold do
    add escapeAngle + i and escapeAngle − i to possibleGoals
end for
sort possibleGoals based on maximum distance the agent can walk in that direction and difference to escapeAngle
return the best of the sorted possibleGoals
```

In Algorithm 3 one can see how to find the best direction to escape to. When that angle is found we use A^* to calculate a path to the node that is in the direction of the angle and the maximum distance away. However only the first three steps of the path are calculated to ensure that the intruder will reconsider the situation it is in after those steps. This makes the evasion route very adaptable for new threats and also for change in the surrounding terrain.

- Deep Reinforcement Learning . For the evasion approach using Deep Reinforcement Learning (DRL), a fully connected feed forward neural network is used. This network has several inputs:
  - Vision. The first thing is the $visionMemory$ of the guard that is considered a threat. If the threat why evasion is performed is a sound then this input will be set to -1. The next inputs are the three most recent and geographically closest $visionMemories$ that this agent has. If those do not exist then the input will again be -1. The $visionMemory$ input consists of the angle to the agent, the distance to the agent, how long ago it was and lastly to which agent type it belongs (guard, intruder).
  - Walls. The x closest walls are also used as an input. A wall is inputted with an angle to the agent and the distance that it is away from the agent. If there are no x closest walls then the input will be -1 for those that are missing.
  - Pheromone. The average pheromone angle.
  - Sound. If the threat why evasion is performed is a sound then the first sound input will be that sound, if it was not then the input is -1. Furthermore, the input contains the two loudest sounds that are currently perceived.
  - Orientation. The current orientation of the agent.

  All inputs are normalized between $[-1, 1]$.
  Since the input size may vary in different situations but the input size to a neural network always has to be the same we used a technique called zero-padding where we pass -1 when a certain input is not there.
  All intruders use the same neural network for training to make it more adaptable for different situations. Furthermore, they receive group rewards and not direct rewards for their own actions. This is to reward working together in the environment. The rewards are given at each time step based on the following heuristic:

Algorithm 4 Group Reward Heuristic
Output: the group reward

```
if all intruders are caught then
    return −25
else if no intruder is caught then
    return +1
else if an intruder escapes a guard then
    return +15
end if
```

2) Capture Target Area: The capture target area task is performed when an intruder is standing in the target area. Since agents generate sound when moving, the intruder wants to move as less as possible in that time. The intruder will only turn $360°$ every few steps to ensure that it will see an incoming guard but it will not move any tiles.

III. Experiments

It is crucial to understand how much the parameters used for sound, visual and marker perception can influence the performance of the agents for surveillance and exploration. Some setting may play an important role on the performance while another may not even have any impact, one of the goals in this section is to have an insight on that. Next to experiments on changing parameters, there will also be some experiments on comparing different type of algorithms used to solve sub problems. Since the exploration and surveillance simulations have different goals, the testings need to be performed differently.

A. Exploration

In addition to the number of agents, which is a constant when performing experiments, the range of the pheromones and their reduction play an important role. These two parameters determine the distance from which they can be detected by another teammate and the reduction per second. Going more into details, the pheromones can be smelled from 10, 20 or 30 tiles away and can get reduced by 0.01, 0.05 or 0.1 per second. The maps used during exploration have two different main characteristics, one has few walls but many small rooms and corridors while the second one doesn't have close spaces but just disconnected walls. The first one tests the ability of the algorithm to find the entrance of the rooms, while the second checks the ability of the guards to reroute the search when interrupted by a wall. To see the maps, see section VII-A. The main goal of this experiment was to understand the effect of using pheromones on the time taken to complete the exploration on the map.

B. Surveillance

For this phase, 4 settings have been triggered, three of them are related to the sound emitted by the agents. As already mentioned in section II-A4, the actions that produce sound are footsteps, rotating and yelling. One of the experiments is testing different distances these sounds can be heard from. For the footstep sound distances of 8, 16 and 24 will be tested, while for rotating a distance of 5 and 10 will be tested. For yelling the distances 30, 50 and 70 will be experimented with. For these experiments a fixed number of guards and intruders will be used, namely three for both. There is also an experiment on changing the number of guards and intruders, for that experiment the sound parameters are fixed, namely the footstep distance being 16, the rotation distance being 5 and the yelling distance being 50. Next to these experiments a Deep Reinforcement Learning (DRL) approach for evasion, see section II-D1, and a neural network for sound deciding, see section II-B3, will be compared to algorithmic approaches. Due to the fact that the performance criteria of the surveillance experiments is the win rate of the guards, different types of maps will have a great impact on the results. Even different spawns will greatly impact the outcome. That is why for all experiments one map is used with four different spawning locations for both the guards and the intruders. Resulting in a total of four changing maps. The maps can be seen in section VII-A. The main goal of this experiment was to find the parameters that would improve performance for surveillance, whilst understanding the effect that the number of agents could have on the perception of the agents. Finally, it was fundamental to compare the DRL agent with algorithmic approach for task decision, as well as the classification approach with the algorithmic approach for sound detection, in order to choose the better performing algorithm.

IV. Results

In this section the results of first the exploration experiments and secondly the surveillance experiments will be presented in the form of graphs.

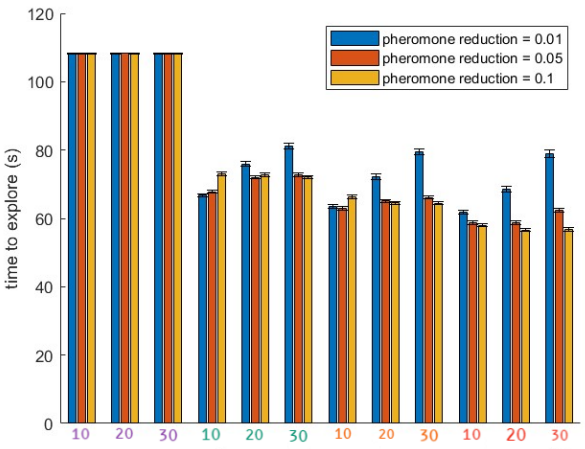

Fig. 4. Exploration map 1, where the number of agents are 1, 3, 4 and 6 from left to right matching the colors respectively and the x axis represents the pheromone's range

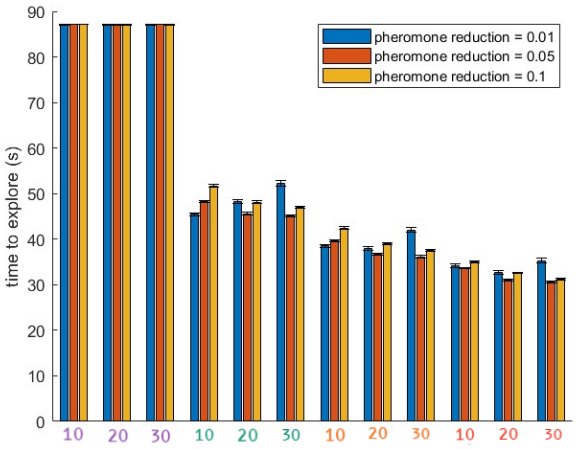

Fig. 5. Exploration map 2, where the number of agents are 1, 3, 4 and 6 from left to right matching the colors respectively and the x axis represents the pheromone's range

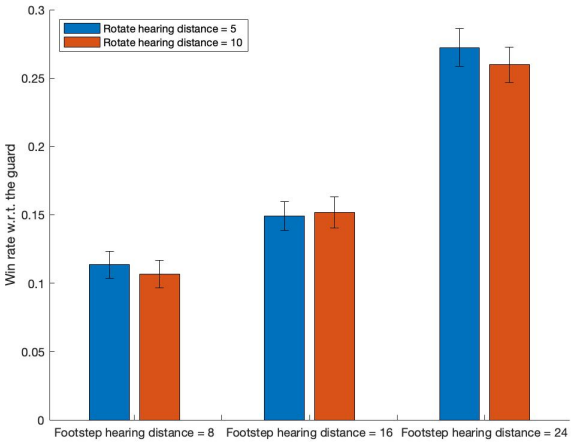

Fig. 6. Surveillance guard yell distance = 30

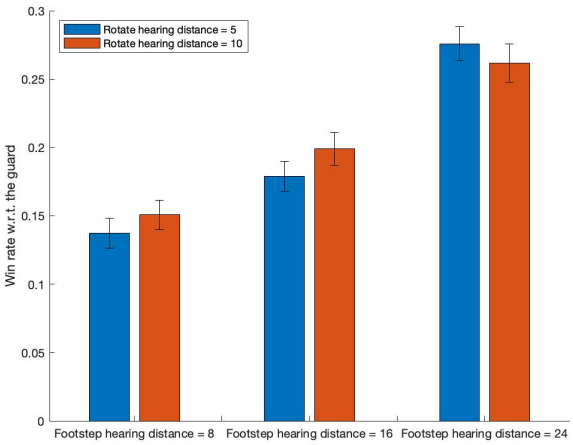

Fig. 7. Surveillance guard yell distance = 50

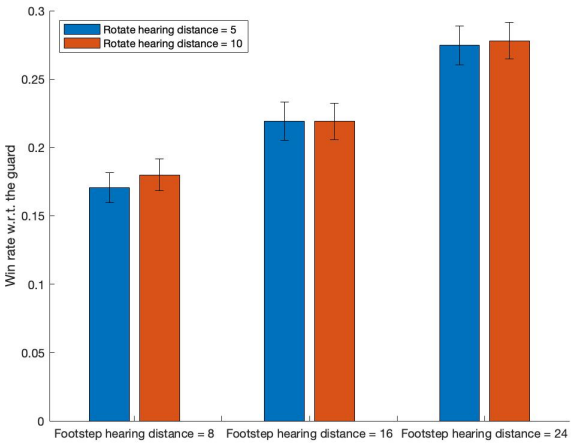

Fig. 8. Surveillance guard yell distance = 70

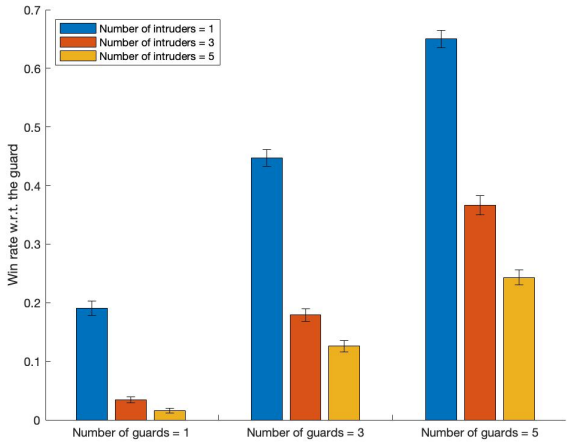

Fig. 9. Surveillance changing number of guards and intruders

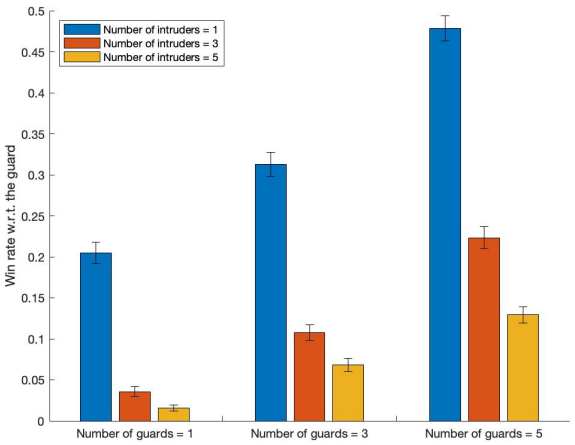

Fig. 10. Surveillance with guards using the sound deciding neural network

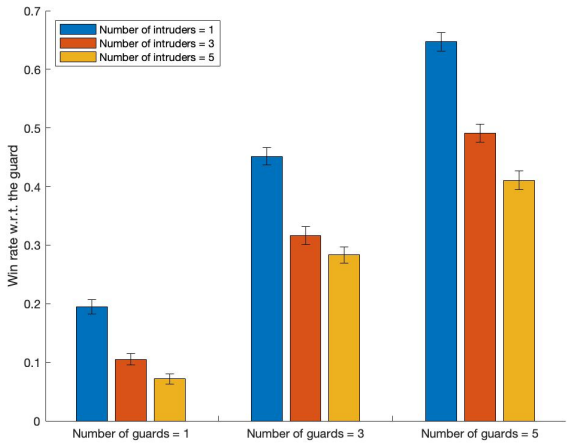

Fig. 11. Surveillance with intruders using the DRL evasion task

V. Discussion

In this section the results, shown in the results section above, will be discussed. Starting with the experiments on the exploration game. Then, we will take a look at the experiments performed on the surveillance game.

A. Exploration

By looking at the plot of the first map Figure 4, many conclusions can be made. First, when using one agent the exploration time is fixed for all the settings, this is because the pheromone doesn't have any impact is there are no other agents to detect it. Second, the 0.01 reduction level, in most cases, seems to be slow down the exploration. The reason of that is because the pheromones last too long on the maps and hence the agents are not able to explore the map efficiently. Third, the yellow and orange bars have very similar values with the former performing slightly better, especially when the search is performed by many agents. The size of the map is relatively small for 6 agents, this is why the fewer the pheromone lasts, the fastest the exploration lasts.

Similar results can be found when using the second map, but, since the map is smaller, the exploration time has an average decrease of 33One of the man differences from the previous graph is the fact that the average time between the settings for the same number of agents is much closer. Since the map doesn't have any rooms, the levels of pheromones have a fewer impact on the final time. Furthermore, a clear decreasing trend on the exploration time can be noticed when increasing the number of agents.

B. Surveillance

First we will take a look at the experiment where we examine the impact of sounds on the winning rates of guards, and thus intruders. Overall it can be observed that in all figures 6, 7 and 8 the win rate of the guards increase when increasing the distance footsteps can be heard from. This means that guards are better at acting on sounds, while they take more advantage of an increasing hearing distance. It can also be seen that increasing the distance to which a rotation can be heard, increases the win rate of the guards. Next to the fact that guards are already better at acting on sounds as seen for the footstep distance, the rotation distance has something more interesting. As explained in section II-D2, when intruders are in the target area, they will each x time steps do a 360° rotation, which obviously causes rotation sound to be produced. When increasing the distance this sound can be heard from, guards are more likely to detect intruders being in the target area, which causes guards to win slightly more games. Looking at changing the distance guard yells can be heard from it can be observed that the win rate for the guards increases when increasing the distance guard yells can be heard from. The reason for this is simply because the greater this distance is, the more agents that are further away will respond to this yell. This makes it easier for guards to catch an intruder. Furthermore, the intruder being chased is high likely close to the target area, while intruders move towards this area. Because of this the chance of catching more intruders, thus the win rate for the guards, increases.

Furthermore, looking into the influence of changing the number of guards and intruders on the win rate of the guards. From figure 9 it can be observed that in general, guards win a lot less than intruders. We think this is mainly due to the fact that intruders get the initial angle of the target area, so they already approximately know in what direction they should go, whereas guards do not have this information. This means that a lot of times intruders are able to win the game without interfering with guards. Another reason could be that intruders are good at evading, while guards are not that great at pursuing. This causes guards to all chase one intruder, because of the guard yell, resulting in free rein for the other intruders.

Moving on to the experiment for comparing the algorithmic evasion task with the DRL evasion task. It can be observed from comparing figure 11 with figure 9 that when there is more than 1 intruder the win rate of the guard increases, thus the win rate of the intruder decreases, when the intruder uses DRL for evasion. Another important observation is that when using only one intruder the win rates stay the same. This indicates that intruders evade a lot of time from their teammates, while this can only happen when there are multiple intruders. It is hard to tell if the DRL performs worse only when evading from teammates leading to unexpected behaviour of the intruders, or also when evading from guards. However, the latter would make the most sense, while the cause of evasion should not have that much of an impact.

Finally, the experiment for comparing the algorithmic and the neural network sound deciding will be discussed. When comparing figure 10 and figure 9 it can be seen that the win rate for the guard decreases. Note that in this experiment only the guard uses the neural network sound deciding model. This is interesting, because that would imply that the neural network model performs worse in game than expected. As stated in the methods section II-B3 the neural network achieved an accuracy on classifying sounds of around 84%. However, remember from the methods section II-B3 that this network is evaluated at each time step for each visual memory. This means that errors will actually occur quite often. Next to this, observe that the win rate for the guards stays the same when only using one guard. This implies that guards are not matching sounds with the vision memory of their teammates, causing them to look after each other a lot of times, and looking at the win rates probably a lot more than they would do during the algorithmic approach. The issue of not matching sounds correctly is that when guards will try to spread out, they will want to go back to see if the sound of their teammates is really coming from their teammates.

VI. Conclusion

In conclusion, the results show that different parameters can positively affect the results of the surveillance. In fact, using pheromones on the exploration and therefore coverage of the map results in an decrease of 30% on the simulation time and improves the performance of the surveillance. Secondly,

indirect communication in the form of sound also improves the performance of the guards. In fact, we have seen that increasing the distance of the guard yells improves the guard performance. Moreover, it was shown that increasing the distance of the sound produced by the intruders also has a positive impact on the performance of the guards. Furthermore, when the number of guards increases, the exploration time always reduces. On average, there is a 25% reduction in time. The same phenomenon occurs during the surveillance phase. Going more into details, when the number of intruders and guards is even, the win rate indicates that the intruders always have an advantage over the guards. Whereas the guard team has the upper hand only with a sufficient number of team members. The experiments showed that the Classification Neural Network approach performs worse than the algorithmic one. This is because when using the former, guards are not matching sounds with the vision memory of their teammates. This causes them to go back to check if the sounds they hear are really coming from their teammates, causing a reduction in performance. The last experiment showed that the DRL approach for evasion performs worse than the algorithmic approach. High likely due to the fact that intruders are evading from other intruders in an unexpected way. Further investigation on how to improve the classification neural network for sound deciding and how to improve the DRL approach would have to be made in order to have the results that were expected.

References

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Nguyen, A. A. (2021). Scalable, decentralized multi-agent reinforcement learning methods inspired by stigmergy and ant colonies. *arXiv preprint arXiv:2105.03546*.

Pramanik, M. I., Lau, R. Y., Yue, W. T., Ye, Y., & Li, C. (2017). Big data analytics for security and criminal investigations. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, *7*(4), e1208.

Quinn, M. (2001). Evolving communication without dedicated communication channels. In *European conference on artificial life* (pp. 357–366).

Rossi, F., Bandyopadhyay, S., Wolf, M., & Pavone, M. (2018). Review of multi-agent algorithms for collective behavior: a structural taxonomy. *IFAC-PapersOnLine*, *51*(12), 112-117. doi: https://doi.org/10.1016/j.ifacol.2018.07.097

VII. Appendices

A. Maps

Legend of the maps:

- Blue tiles are spawns for the guards
- Red tiles are spawns for the intruders
- Green tiles are target areas

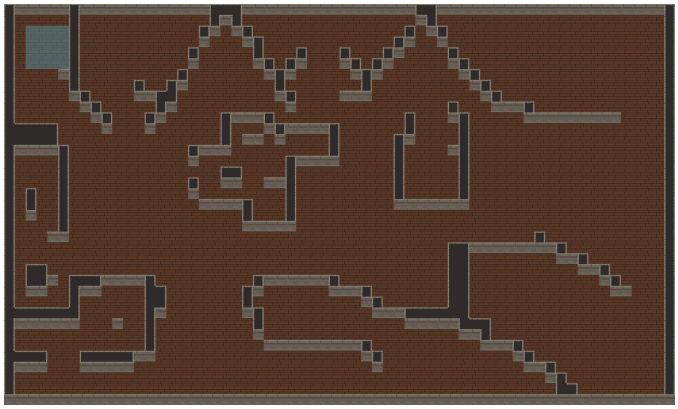

Fig. 12. Exploration map 1

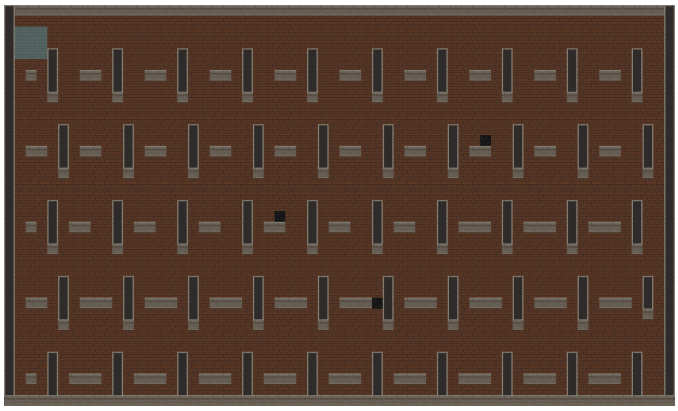

Fig. 13. Exploration map 2

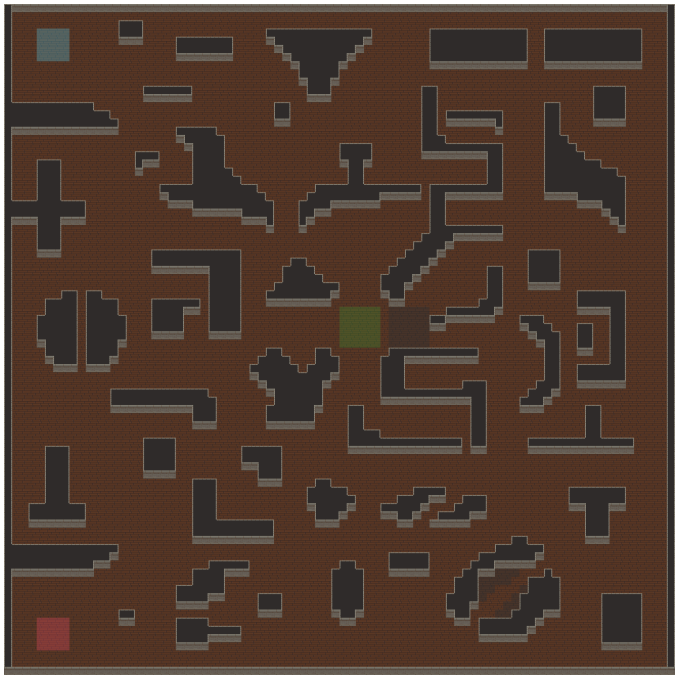

Fig. 14. Surveillance map 1

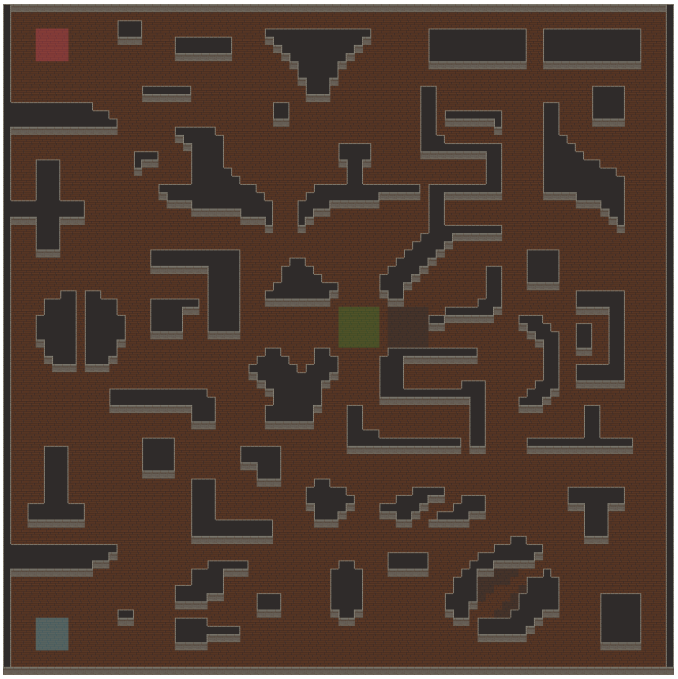

Fig. 15. Surveillance map 2

Fig. 16. Surveillance map 3

Fig. 17. Surveillance map 4