ID2209 - Distributed Artificial Intelligence and Intelligent Agents

Final Project – Behavior of different agents

Group 12

Alexandros Nicolaou

Alexandre Justo Miro

24 December 2019

Contents

Introduction

In this assignment, we were tasked with completing four tasks:

1. Create the basic model to combine our previous knowledge of gama applications (like utility etc).
2. Demonstrate an algorithm using the BDI architecture
3. Demonstrate an algorithm using reinforcement learning
4. Creative model

How to run

Run GAMA 1.8 and import ‘Basic Model.gaml’ as a new project. Press ‘BasicModel’ to run the simulation. The same applies for the ‘Challenge Model 1.gaml’, ‘Challenge Model 2.gaml’ and ‘Creative Model.gaml’, with the corresponding buttons ‘ChallengeModel1’, ‘ChallengeModel2’ and ‘CreativeModel’.

Approach

To demonstrate the basic model we used our knowledge and experience from the previous assignments such as utility theory, graphs etc. For the first challenge we followed some tutorials of the gama documentation and some examples in the literature also helped us to demonstrate the BDI. For the second challenge however there was no literature about reinforcement learning in gama, and we don’t have any experience in this. For that reason we read the literature that was provided in the presentation of the assignment, and imitated the coding structure of other programming language to demonstrate this. For the creative part, all we needed is our imagination.

Experiments and Results

Several experiments were carried out. Among them, the Basic Model, where the scenario, agents, rules of interaction, and useful graphs were built. Other additional experiments were built on top of the Basic Model. These were: Challenge 1, where Belief-Desire-Intentions behavior was implemented; Challenge 2, where Reinforced Learning was implemented; and Creative Idea, where a creative idea of our own was implemented. Description of these experiments and the corresponding results follow.

Basic Model

The basic model was built with species “place” and “guest”.

The first one, “place”, consists of static points where guests can meet. There are three different types of place: “Bar”, “Stage”, and “Terrace”. A type is randomly assigned to each “place” agent. The parameter “number_of_places” controls the total number of places in the simulation, and is

by default set to 9. The location of places are manually set to be equally distributed in the 100x100m square.

The second one, “guest”, consists of agents with the “moving” skill. The parameter “number_of_guests” controls the total number of guests in the simulation, and is by default set to 50.

Each guest has 3 personal traits: “generous”, “friendly”, and “wealthy”. These are assigned a random integer value between 0 and 10 each. If the value is greater or equal than 5, the agent is considered to be, for instance, generous, and will be prone to accept certain kinds of proposal; however, if the value is lower than 5, the agent will directly reject certain kinds of proposal. This will be later explained in more detail.

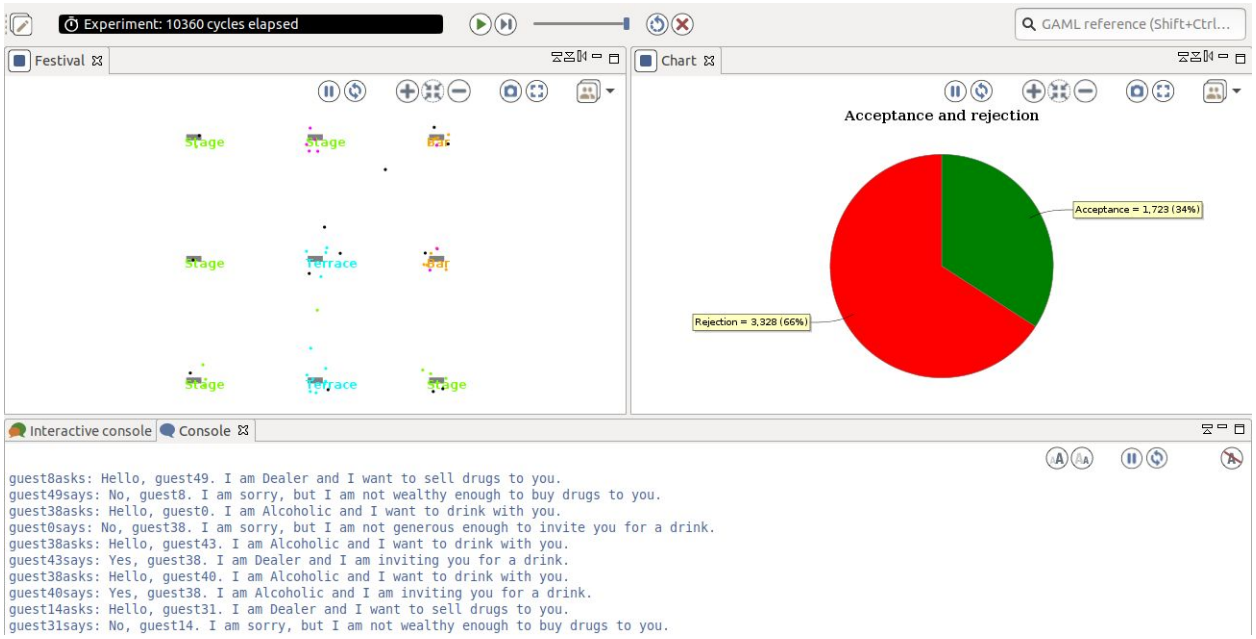
Aside from the aforementioned personal traits, each guest is randomly assigned a “type” of the following 5: “Party”, “Chill”, “Alcoholic”, “Instagrammer”, and (drug) “Dealer”. These types affect directly how guests interact between them and where they prefer to go. Guests assigned with “Party” type (color green) are always seeking for a place of type “Stage”, guests assigned with “Chill” type (color cyan) are always seeking for a place of type “Terrace”, and guests assigned with “Alcoholic” type (color orange) are always seeking for a place of type “Bar”. Guests assigned with “Instagrammer” (color magenta) and “Dealer” (color black) types do not have any preferences and go randomly to any places of whatever type.

Once the agents have chosen which place to go, they have one, and only one chance to do proposals to all other guests that are within a distance of 5m. They do that by means of an “ask”

statement, either in their way to the place or once they have already reached the place. After a limiting time named “autonomy”, which randomly takes a value between 200 and 400 cycles and is an attribute of each guest, has passed; the target point changes and they have one more chance to do proposals again, the time counter is restarted, and so on. This limit was set since, otherwise, they could be continuously asking the same guest instead of asking only once and receiving an answer only once and stop insisting.

Each guest has a set of rules. These rules define how they interact with other guests. In the end, it all comes to accept or reject proposals. Each guest features 5 attributes, called “interact_party”, “interact_chill”, “interact_alcoholic”, interact_instagrammer”, and “interact_dealer”. They take a float value between 0.0 and 1.0, and define how a guest reacts to the proposal of a guest of type “Party”, “Chill”, “Alcoholic”, “Instagrammer”, or “Dealer”; respectively. For instance, a guest of type “Party” has an attribute “interact_party” equal to 1.0, since it will always accept the proposal of another guest of type “Party” to dance; or a guest of type “Chill” has an attribute “interact_party” equal to 0.0, since they will never accept the proposal of another guest of type “Party” to dance. This value between 0.0 and 1.0 of the different attributes is the argument of a “flip” operator, which yields a “true” or “false” bool variable that ultimately means the acceptance or rejection of the proposal.

On top of that, the 3 personal traits affect these rules. For instance, if the value of “friendly” is lower than 5, a proposal to dance coming from a “Party” guest will always be rejected, indifferently of the algorithm explained in the paragraph above.



The simulation is continuously running and a pie chart is displayed. It shows the proportion of accepted (green) and rejected (red) proposals and monitors the number of accepted and rejected proposals. These are automatically updated while the simulation runs.

Challenge 1: Belief-Desire-Intentions

For the first challenge of BDI, we used 6 agents:

- Guest: The guests are the people that attended the music festival. As in every music festival, guests pass through many conditions while they are in the festival. Either they want to drink, dance, pee, do drugs, get money, or leave. Thus, guests are wandering around till a need is present. However, depending on the occasion, one need may be more important for the agent than other. This can be illustrated in the gama agent using the BDI architecture. The user can change the number of the guests by changing the **number_of_guests** variable. 50 is the default number.
- Drug Dealer: Drug dealers consider music festivals as an opportunity for them to become rich, since many guests are interested in doing drugs there. They are allocated a random amount of drugs, and when they run out of them because the sold everything, they go to leave the festival by going to the exit. Their size indicates the number of drugs they still have. Their location is hidden and fixed, and the interested guests visit them. The user can change the number of drug dealers by changing the **number_of_drugdealers** variable. 6 is the default number.
- Places: There are 4 main places that the guests can visit named:
    - Bar
    - Bank
    - Toilet
    - Exit

  Their location is fixed at the places **bar_location, bank_location, exit_location and toilet_location**. The user can also change the number of drug dealers by changing the number_of_bar, number_of_bank, number_of_exit and number_of_toilet variable. 1 is the default number.

Predicates

In order to perform the BDI architecture, in gama we have to define predicates that we will use later. The **predicates** in our case will be:

- dance
- do_drugs
- drink
- get_money
- need_pee
- pee
- go_back
- need_money
- Nirvana

Then we need to define the desires of the guests. **The initial desire is dance.**

Guests pay an amount every time the buy drink or drugs. Then, when the amount of their money fall below a threshold level, the belief that they need more money is activated with: **do add_belief(need_money).**

Perceive

For the purpose of this simulation, we only need one perceive, and this is that whenever a guest pass near a drug dealer, and he is already drunk (passed a drink threshold), then he will probably want to move on to the next level which is to take drugs. Therefore a belief called **nirvana** is being added.

```
// Perceive
perceive target: drug_dealer where (each.drug_quantity > 0) in: view_dist{
    if myself.drunk_level >myself.drunk_threshold {
        drugs_at_location<-self.location;
        ask myself {
            do add_belief(nirvana);
        }
    }
}
```

Rules

Then we need to define the rules of the BDI. The strongest rule with strength of 10, is that when a guest runs out of money, then a **get_money** desire will be activated since without money he can't buy anything. Then in case an agent reaches the nirvana condition, his desire is equally strong between to **do_drugs** or **drink**. The least strong rule is the desire of the guest to **pee** since he/she can hold on. Entertainment and party is above everything in a festival.

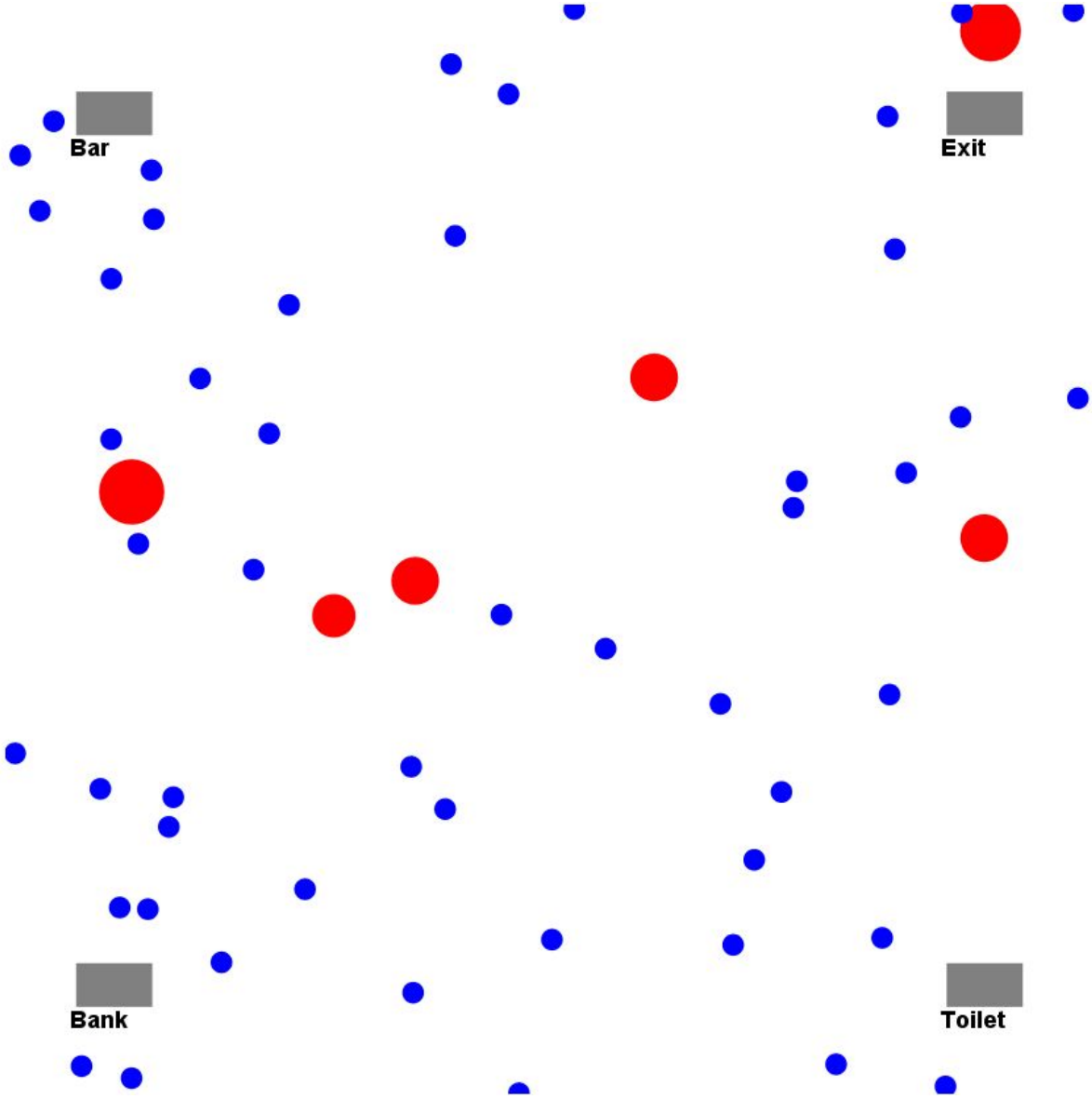
```
// Rules
rule belief: need_money new_desire: get_money strength: 10.0;
rule belief: nirvana new_desire: do_drugs strength: 8.0;
rule belief: nirvana new_desire: drink strength: 8.0;
rule belief: need_pee new_desire: pee strength: 1.0;
```

Plans

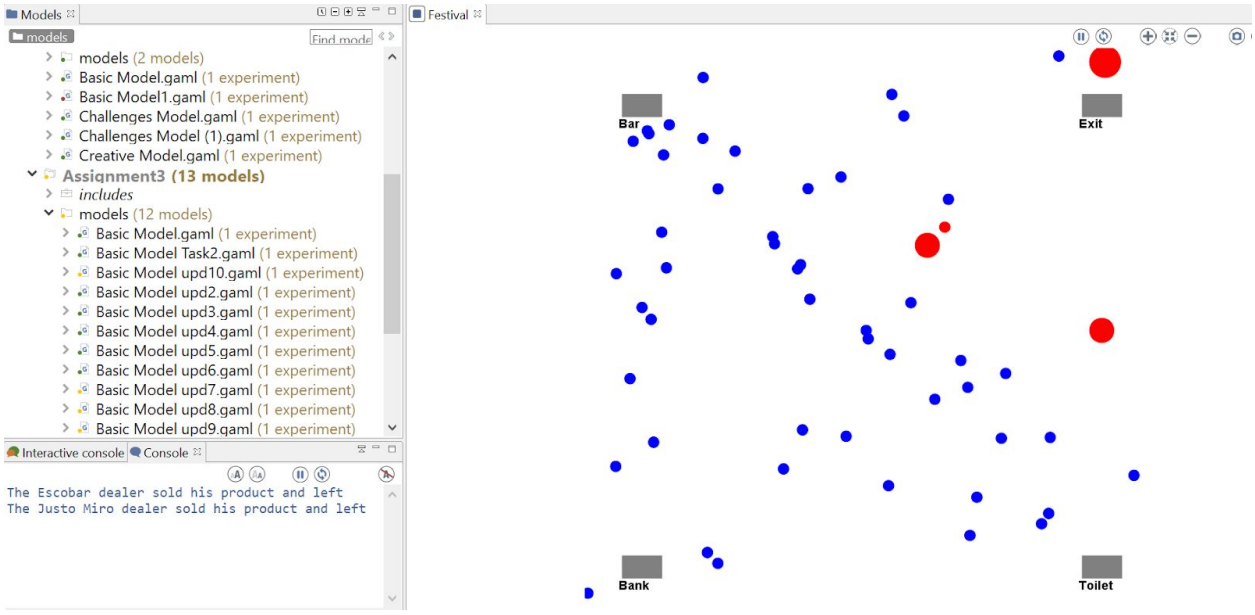
The last part of the simulation is to add the plans. Based on the beliefs and desires, guests will choose to perform these plans:

- Dance: They just wandering around. They thirst level is increasing while they are dancing. While the thirst level is below a certain amount, they have the desire to drink.
- Go back: This plan is activated when guests either went for a drink, to the bank or to seek for drugs. Then they return to their initial location to continue dancing.
- Get Money: In that case, guests go to the bank that is located in the festival to get more money.

- Drink: Guests go to the bar to get a drink.
- Drugs: When guests believe there is a drug dealer close to them, they approach him, buy and do drugs.
- Pee: Whenever is needed, guests go to the toilet to pee.



In this image, with blue are the guests, red are the drug dealers in random locations, and grey are the 4 different places.



Whereas in this image we see that two drug dealers already sold their drugs and left the festival, as is also indicated in the left console. Also the size of the rest drug dealers was decreased due to the fact that they have sold some of their drugs.

We also observe most of guests being around the bar, and some of them heading towards the toilet since it is a need that is always active (but not strong).

Requirements for passing this challenge is to clearly demonstrate (in your presentation and paper) Belief-Desire-Intentions (BDI) behaviour in agents.

https://gama-platform.github.io/wiki/BDIAgents

Challenge 2: Reinforcement Learning

The challenge 2 of reinforcement learning used 4 agents named:

- Guest

Guest are the people attended the festival. In every music festival one can find every type of guests, from music-dance lovers, guests that seek food, alcoholic guests, and more often drug addicts since the festival seems to be an attractive place for them to take their dose. The guest in our case is a drug addict and his goal is to chase and catch every drug pill in the festival, without being caught by the police. The guest moves one grid at a time. He/she doesn't know the location of the drugs, so he/she has to learn the location by trial and error. The user can change the number of the guests by changing the **number_of_guests** variable. 1 is the default number.

- Grid

The problem was based on a grid layer similar to the Nqueens problem. Thus, a grid environment representing the music festival was chosen. The choice of grid was simply to make the problem more visually distinguishable. The user can change the grid size by changing the **grid_size** variable. 8 is the default number.

- Drugs

The drugs are being placed in random locations around the music festival. Their location is not known, otherwise the police would know and would easily catch the drug addict. The user can change the number of drugs by changing the **number_of_drugs** variable. 6 is the default number. Every drug is different from each other, thus rewards the drug addict differently based on its size. The size of the drugs can be modified by changing the **drug_reward** variable. In our case the drug_reward variable is a random integer in the range of 1 to 10.

- Police

As in every festival, police is suspected for the drug movement. Thus, there is a stand-by police officer ready to chase the drug addict before he/she captures all of the drugs. The police officer is not assigned, a grid location, but moves free in the festival, with targetpoint the drug addict location. The user can change the number of police officers by changing the **number_of_police** variable. 1 is the default number.

The way the guest learn the location of the drugs, is using reinforcement learning. For this, these variables had been used as default, however the user can change them accordingly:

- total_episodes: 15000
- learning_rate: 0.8
- max_steps: 99
- learn_step: 100
- discount_rate: 0.9

And also the exploration parameters:

- epsilon: 100.0
- max_epsilon: 100.0
- min_epsilon: 1.0
- decay_rate: 0.005

In this case, the Q table for reinforcement learning was being filled based on the action that the guest chose in every state. **The states represent the index of the grid the guest is currently on, whereas the actions represent the 4 different moves that the guest can do: left, right, up and down.**

In every episode, the guest performs max_steps (99) steps, and then he respawns to perform the next episode. The learning process is completed after total_episodes (15000) episodes.

For filling the Q table, we used the rewards approach of reinforcement learning. Thus, the quest can be rewarded these:

- **Random positive reward equal to the drug_reward variable, if the guest finds the location of the drug.**
- **No reward, when the guest is located in a non-drug grid. A user can make this reward negative to help the guest learn faster.**
- **Negative reward equal to -10 when the guest is located at the same grid that the police is. The negative reward also help the guest to learn faster.**

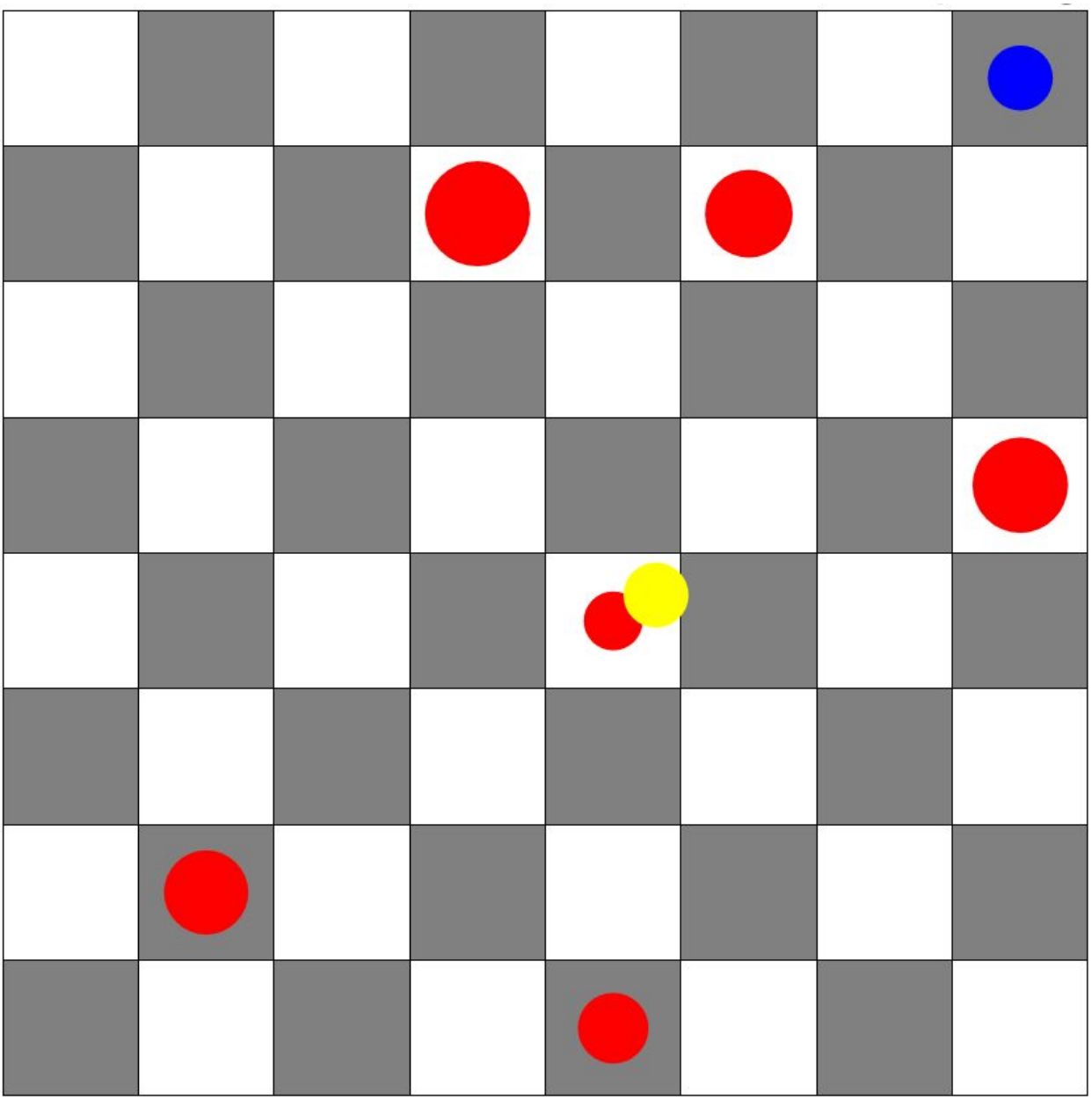
In every step, the guest chooses to either perform the best action based on the Q table, either performs a random action for the sake of further exploring the festival.

When the guest completed all the episodes, then he performs the best road based on what he learnt using the reinforcement learning parameters. The optimized road must therefore consist of as small as possible number of grids, in which some of them are the grids where the drugs are located.

For every state, i.e. grid, the agent picks the action with the higher Q table value. If this action is not available then he chooses the second highest action's value. In case of all the values of the actions of a state, are equal to 0.0, which occurs when the guest is far from every drug, then the guest picks a random action.

The whole process finishes when the guest visited and captured every drug in the festival. Then the program prints the optimized path in the form of grids.

In the following images, the reinforcement learning process is presented:



The guest is being shown as blue, the drugs red, and the police as a yellow dot. This is the starting position. The guest then explores the grid until he finds the location of every drug. As he moves, a console informs us of the moves he chooses.

```
The guest moves to his down grid
The guest moves to his up grid
The guest moves to his down grid
The guest moves to his left grid
The guest moves to his left grid
The guest moves to his right grid
The guest moves to his right grid
The guest moves to his left grid
The guest moves to his right grid
The guest moves to his up grid
The guest moves to his down grid
The guest moves to his left grid
The guest moves to his right grid
The guest moves to his left grid
The guest moves to his up grid
The guest moves to his up grid
The guest moves to his left grid
The guest moves to his right grid
```

The final Q table, in which we can see the preference of the agent, for each state to choose the action that will lead him to the drugs, is:

The Q table is
[[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[432.8229306436532,435.3288450387855,0.0
,486.6999456582233],[0.0,0.0,0.0,0.0],[320.53418319463907,320.48238693917443,0.0,356.16
15203794428],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[329.42301008
615976,366.02587758293873,307.36724564610984,314.36784580742057],[0.0,0.0,0.0,0.0],[80
4.6297206718623,792.0539718607927,720.683475162201,721.1107496450846],[0.0,0.0,0.0,0.
0],[357.7730329041882,321.9955724768902,321.9712684821902,321.9953645358462],[0.0,0.
0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[379.66488517602255,384.2709466
4680044,440.8227350638519,345.4371441860904],[0.0,0.0,0.0,0.0],[321.83007881870446,321
.6817494262001,357.6384200794815,321.7938449237789],[0.0,0.0,0.0,0.0],[378.17640427830
275,0.0,378.24497059893696,420.41694035234957],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0
.0,0.0],[0.0,0.0,0.0,0.0],[112.39450813269868,112.35069172675436,112.41153234599486,125.
16216303112743],[0.0,0.0,0.0,0.0],[377.9376616000308,419.931056770345,377.90142221023,
377.7465819339315],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[111.155
67077531996,123.93775082313987,107.56561605692563,110.33952912872819],[0.0,0.0,0.0,0
.0],[125.65712543895299,112.98387107779926,112.96435528453895,112.57063850791982],[
0.0,0.0,0.0,0.0],[378.8181034930215,0.0,420.9431695366734,378.82899141232645],[0.0,0.0,0.
0,0.0],[274.39376725839554,270.5061280234683,266.7202346018606,309.11791528916604],[
0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[112.97780770154212,113.05666924556525,125.73200215130
22,112.59001632611381],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,318.542057118
6894,284.5591499631798,280.0167275102745],[0.0,0.0,0.0,0.0],[323.45325341211685,291.10
792739232687,290.9916493607283,291.04146224174093],[0.0,0.0,0.0,0.0],[198.80834131395
437,198.78318646530727,198.74869159069735,220.9266099653073],[0.0,0.0,0.0,0.0],[0.0,0.0,

0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[295.6136822153415,295.6143737831155,328.460603
88409904,0.0],[0.0,0.0,0.0,0.0],[198.24737360226558,220.48028373565222,198.41378740071
357,0.0],[0.0,0.0,0.0,0.0],[220.48436892256757,198.25788775834255,198.41193871934203,0.
0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]]

When the agent performs learning, he finally computes the optimized path and follows it to capture the drugs. When the process is finished, a message is being shown in the console “The final road is…”. In this case, the best path is:

The final road is
[my_grid(36),my_grid(28),my_grid(20),my_grid(12),my_grid(11),my_grid(3),my_grid(4),my_grid(
12),my_grid(13),my_grid(5),my_grid(6),my_grid(5),my_grid(13),my_grid(21),my_grid(29),my_gri
d(28),my_grid(27),my_grid(19),my_grid(18),my_grid(17),my_grid(16),my_grid(17),my_grid(16),
my_grid(8),my_grid(0),my_grid(8),my_grid(9),my_grid(10),my_grid(2),my_grid(3),my_grid(2),my
_grid(3),my_grid(4),my_grid(12),my_grid(11),my_grid(19),my_grid(18),my_grid(17),my_grid(16),
my_grid(24),my_grid(32),my_grid(33),my_grid(41),my_grid(49),my_grid(57),my_grid(58),my_gri
d(57),my_grid(56),my_grid(57),my_grid(58),my_grid(57),my_grid(58),my_grid(50),my_grid(51),
my_grid(43),my_grid(51),my_grid(43),my_grid(51),my_grid(52),my_grid(60),my_grid(52),my_gri
d(53),my_grid(61),my_grid(62),my_grid(63),my_grid(62),my_grid(61),my_grid(60),my_grid(61),
my_grid(53),my_grid(45),my_grid(46),my_grid(38),my_grid(46),my_grid(38),my_grid(39),my_gri
d(31)]

The path doesn’t seem to be optimal, however the agent probably learned from the previous episodes so that he managed to capture all the 6 drugs, in around 70 moves, in a 64 grid, without the police managed to catch him. One, can further improve the algorithm by altering the reinforcement learning parameters.

In order to improve the code, the next steps would be to find a suitable visual way to show the final path of the agent, but also the Q table, so that to make it easier for a user to follow the path.

The algorithm can also be improved by training further, or introducing neural networks and making it deep reinforcement learning.

Another reason of not optimal reinforcement learning algorithm is the fact that it is the first time we learned and tried it. We could definitely improve it in the future, when we will have more experience in it.

Last, the rewards can be also modified so that the agent gets a penalty for as long as he is on the road, but we didn't have time to properly introduce this feature. Now the agent does not get a reward if he doesn't find drugs.

Creative Idea

Software tools that allow to work with agents, like Gama, can be very useful for serious applications such as medicine. In particular, the main field this Creative Idea is related to is the spread of infections. However, since that is an unpleasant thing to do, a funny game of infection was instead created.

The basic structure of the project is kept, but now the different types of agents (Party, Chill, etc.) are gathered in teams. This is, there are 5 teams, called Team Party, Team Chill, etc. A new agent, called Instructor, appears in scene. It is the agent responsible for starting the game and communicating what is happening in real time via FIPA.

The Instructor picks one member of each team. They are going to be the first infected agents. So at the beginning there are 5 infected agents, 1 per team. They keep interacting in the festival as in the Basic Model, as if the infection did not exist. However, when one agent accepts the proposal of an infected agent, it becomes infected as well.

Although been infected, agents are still allowed to participate. However, when all the members of one team are infected, the Instructor communicates this to the members of the team. The whole team is eliminated and they disappear from the game thanks to the “do die” statement.

Teams keep getting eliminated until there is only one team standing. That last team to survive are declared the winners of the infection game.

Following a screenshot of the middlegame. All 5 teams still remain in the game. The agents infected took the appearance of a skull of the color of their team. The Instructor is displayed in the upper part of the board. A scoreboard helps to see which teams are doing better.

Following a screenshot of Team Dealer after winning the game. Their appearance turned into the appearance of a trophy of the same color of their team.

Qualitative / Quantitative questions	*Answer*
Time spent on finding and developing the creative part	12 hours
In what area is your idea mostly related to...	Infections
On the scale of 1-5, how much did the extra feature add to the assignment?	4
On the scale of 1-5, how much did you learn from implementing your feature?	3

Discussion and Conclusion

The final project consisted of both easy and difficult tasks. Besides we didn’t aim of managing to take all the bonus points, in the end we tried all the tasks (challenges included), because that way we had to gain some valuable knowledge that we will probably need in the future such as BDI and reinforcement learning. We could definitely have done better with our algorithms, however for beginners in gama language and AI we are satisfied with the level we reached. We also may improve our codes after we deliver it, so that we can make use of them in future applications. Last, the fact that there roughly is any documentation for gama language and especially reinforcement learning, motivated us and pushed us to be one of the first that will create robust algorithms. They journey continues...