	**Document:**	IMAS practical work
	Course:	ETSE-URV, 2023-24

Table of contents

Practice description

The main goal of the practical work of this course is to put in practice the basic concepts of agent technology in teams of 4 or 5 students. We will use Dedale[1], a JADE-based agent environment framework.

The system aims to simulate the treasure hunt problem. The scenario includes several types of agents, which will have to cooperate and coordinate to explore an unknown map. The goal is to optimize the picking of the treasures that can be found on it.

The environment will be an undirected graph. This graph represents the map in which the agents will be able to move. Each of the nodes, which has a unique identifier associated, is a place on the map. They only allow one agent at a time, and can contain treasures. Edges on the graph are the paths the agents can take to travel between nodes on the map.

You will use the data from OpenStreetMap to create your own map. The map should have at least 500 nodes. Once you have downloaded and modified the map, it must be converted to DGS format using gs-geography project. To create it, follow the same steps as we saw on the second lab session.

Treasures in the map contain a determined amount of either gold or diamonds. They are stored within a safe, which is locked. In order to unlock it, a minimum required expertise in strength and lock-picking is needed. If several agents are near the safe, their expertises are automatically aggregated. Once it has been unlocked, any agent with enough storage capacity can grab all, or part of the treasure.

When the system starts, all the agents will be placed on the map. The types of agents available for this scenario are the following:

1. **Explorer**: its goal is to explore the environment. It is unable to grab treasures, but has the best strength and lock-picking abilities. It can share their known part of the map with other explorers to gradually construct the map of the whole environment. It can also compute the path from a given node A to a given node B if they are on its map.
2. **Tanker**: its goal is to carry the obtained treasures, both gold and diamonds. Even though it can carry a nearly unlimited number of treasures, it is unable to collect treasures and to unlock safes.
3. **Collector**: its goal is to collect the treasures. It has a limited capacity backpack to carry either gold or diamonds, just one type of treasure. It can grab as many treasures as its backpack allows. If a Tanker agent is in range, it can drop the treasure of its backpack to the Tanker. It has limited abilities to open safes.

The final multi-agent system will use the collaboration between the different types of agents to grab all the treasures in the minimum steps as possible. As the map is unknown at the beginning and not all the agents can unlock and carry all the treasures, a coordination mechanism between the agents is needed to complete the treasure hunt.

[1] https://gitlab.com/dedale/dedale-etu

Requirements

The multi-agent system is open to include different design decisions. However, the system must fulfil a list of minimum requirements to accept the practical work:

1. You must create your own map using OpenStreetMap data. The treasures you must add dispersed through the map are the following ones.

Treasure type	Amount	Lock-picking	Strength
Gold	100	3	3
Gold	80	2	2
Gold	60	2	1
Gold	60	1	2
Gold	40	1	1
Diamonds	100	3	3
Diamonds	80	2	2
Diamonds	60	2	1
Diamonds	60	1	2
Diamonds	40	1	1

2. The project must be built and executed using Maven. A README file must also be included with brief instructions on how to build and execute the project.
3. Some statistical information about the system simulation has to be provided. Those include:
    a. Amount of obtained gold
    b. Amount of obtained diamonds
    c. Number of movements per agent needed to complete the treasure hunt
4. The number and characteristics of the agents that must be created, are the following ones:

Agent type	Backpack Capacity Gold	Backpack Capacity Diamonds	Detection Radius	Lock-picking	Strength
Explorer	-1	-1	0	2	3
Explorer	-1	-1	0	3	2
Explorer	-1	-1	0	3	2
Collector	50	-1	0	1	1
Collector	50	-1	0	1	1
Collector	-1	50	0	1	1
Collector	-1	50	0	1	1
Tanker	400	400	0	0	0
Tanker	400	400	0	0	0

Additional agents to facilitate the coordination among the different types of agents can also be added. To communicate, the agents must use the Dedale sendMessage method instead of the JADE one (see provided examples on the Dedale repository). The **communication range** for all agents must be the same. It can be chosen according to the map size and it should be minimised as much as possible. A starting point for the project could be a range of a 100 nodes. Regarding the interaction protocols, all protocols available in JADE can be used.

When the system starts, all the agents will be placed in the map in random nodes ("free"). The system should provide a coordination mechanism for the agents to hunt all the treasures.

It is mandatory to document all changes on the practical work using a Git server. You can use GitHub (https://github.com/github), GitLab (https://about.gitlab.com/) or Bitbucket (https://bitbucket.org/). The teacher must have access to the repository to check the evolution of the project. Use the following email address to add the teacher to the Git repository: jordi.pascual@urv.cat.

Tasks

The first step of the work is the design of the multi-agent system. The design is submitted in task 1 as detailed below, and the lecturer will provide you feedback about it to know the good and weak points. After getting this feedback, you can redesign any component if needed, and then start with the implementation of the multi-agent system (task 2). First you will have to make a basic implementation, and then you will have to think about the best coordination mechanisms (task 3). After that, you will have to implement the final system (task 4).

1. Design activity

Description: In this activity, a first analysis of the practical exercise must be performed. The characteristics of the multi-agent system must be defined. They include the types and properties of the agents in the multi-agent system (according to the studied taxonomy) and the abstract architecture of each agent must be formalised. Those also include their assigned tasks and their utility functions. The overall architecture of the multi-agent system must also be defined. You must use the agent types described in this document, but you can also consider adding additional coordination agents at different levels, although those agents must not have any treasure hunt ability (open treasures or grab them). Because two agents in adjacent nodes cannot swap their positions, agents might encounter a deadlock problem. A solution for this problem must also be proposed. All your design choices must be explained and justified.

Delivery content: PDF report with the characteristics of the agents and your proposed solution for the deadlock problem, and PDF with about 7-8 slides for a presentation of up to 8 minutes.

Delivery deadline: 05/11/2023

Grade weight: documentation and oral presentation 10%

2. Implementation – preliminary revision

Description: The main goal of this activity is to begin the implementation of the practical work. The first set of basic functionality such as the initialization of the system, creation and configuration of the agents, and basic movements of the agents is required.

Delivery content: You do not need to deliver any file. The lecturer will check the implementation in the laboratory class.

Delivery deadline: the revision will be done in the lab in 29/11/2023

Grade weight: 5%

3. Coordination design

Description: The cooperation and coordination mechanisms that will be used to solve the treasure hunt problem have to be designed. They should solve the treasure hunt as efficiently as possible. That is, they should minimise the time needed to grab all the treasures, the amount of movements each agent makes, and use the lowest possible communication range. Write a detailed explanation of the coordination mechanism, and the justification for the decisions you take.

Delivery content: PDF report with the characteristics of the agents and your proposed cooperation and coordination mechanisms for the MAS, and PDF with about 7-8 slides for a presentation of up to 8 minutes.

Delivery deadline: 10/12/2023

Grade weight: documentation and oral presentation 10%

4. Final documentation and implementation

Description: In this activity, the complete implementation of the practical work must be delivered. All the required agents must be implemented, as well as the necessary cooperation and coordination mechanisms among the agents.

A README file must be included with brief instructions on how to execute the project. The configuration files for the environment and agents must also be included.

During the presentation, a demo showing how the multi-agent system works, must be shown.

Delivery content: a PDF report with the details of the multi-agent system design and development, including the obtained results; PDF with 10-12 slides for a presentation of up to 12 minutes and the source code of the project in a ZIP file.

Delivery deadline: 14/01/2024

Grade weight: 20% documentation and oral presentation + 20% implementation

5. Teamwork

Team work will be evaluated too, as one of the advantages of multi-agent systems is to facilitate working in a distributed way. The team must write minutes of the meetings of the group using the forum task available in the URV Virtual Campus. The minutes must indicate how the work is planned and distributed into the different team members, re-planning actions, changes introduced during the project and any other important issue and decisions related with the group work. They are open to include other additional information, such as members of the team present in the meeting, date and location of the meeting, etc. The minutes forum is only accessible by the group and the lecturers.

You can assign different roles in the team if you want: person in charge of the minutes (i.e., secretary), person in charge of reporting, leader, etc.

If someone abandons the course, please write it in the minutes, but also email the lecturers as soon as possible.

Grade weight: 5%

Useful links and recommended readings

As you work in a team, it is useful to use well-known and widely used tools such as:

- As IDE, IntelliJ (https://www.jetbrains.com/idea/)
- JDK 18 or newer (https://jdk.java.net/18/).
- It is mandatory the use of Maven (https://maven.apache.org/) to create the workspace.
- It is highly recommended to use a logging library to trace all the agents. Following, two alternatives are listed:
    - The Apache Log4j 2 (https://logging.apache.org/log4j/2.x/).
    - The Java Logger (https://examples.javacodegeeks.com/core-java/util/logging/java-util-logging-example/).
- As the multi-agent system environment framework, Dedale (https://dedale.gitlab.io/page/dedale/presentation/ https://gitlab.com/dedale/dedale-etu), which is based on JADE (https://jade.tilab.com/).