UNIVERSITAT ROVIRA i VIRGILI

Final Project: Treasure Hunt Problem

Introduction to Multi-Agent Systems (MAI)

Professor
PASCUAL, Jordi

Team members
BEJARANO SEPULVEDA, Edison Jair
edison.bejarano@estudiantat.upc.edu
CARRASQUILLA FORTES, Adrià
adria.carrasquilla@estudiantat.upc.edu
GUENDOUZ, Wafaa
wafaa.guendouz@estudiantat.upc.edu
URCELAY GANZABAL, Lucia
lucia.urcelay@estudiantat.upc.edu
ROSAS OTERO, Mario
mario.rosas@estudiantat.upc.edu
SÁNCHEZ PATIÑO, Natalia
natalia.de.los.angeles.sanchez@estudiantat.upc.edu

Universitat Rovira i Virgili
Avinguda dels Països Catalans, 26, 43007 Tarragona, Spain
Master's degree in Artificial Intelligence
Tarragona, 2022

Contents

Abstract

In this project, we propose a solution to the treasure hunt problem using an intelligent multi-agent system. The problem consists of finding and collecting treasures in a given graph map. To solve this problem, we introduce three types of agents: explorer agents, collector agents, and tanker agents. Explorer agents are responsible for observing the map and gathering information about the location of the treasures. Collector agents collect the treasures and bring them back to the tanker agents, which are responsible for storing the collected treasures.

We use Java and JADE (Java Agent DEvelopment Framework) to implement the intelligent multi-agent system. The system design is intended to be efficient and fast in solving the treasure hunt problem. In this document we gather all the material generated during the different phases of the solution of the problem. From the intial problem analysys, to the final implementation and updates done to improve our solution. We aim to provide with our descriptions a clear understanding of the problem space and how each agent interacts with one another.

Our solution aims to improve the performance of the base methods used for solving the treasure hunt problem by using intelligent agents that can work together to accomplish a common goal. This kind of approaches have the potential to be applied to other similar problems in various domains, such as search and rescue operations, environmental monitoring, and logistics.

Keywords: Treasure hunt, Intelligent Multi Agent System, Java, Jade

1 Introduction

Multi-agent systems are composed of basic units that we call agents. They are computational systems that seek to help us solve different tasks autonomously. The existence of agents has followed the trend in the development of computational systems that is focused on ubiquity, interconnection, intelligence, delegation and human-oriented programming. The use of agents has innovated the way in which systems interact with each other and with us.

Some important properties of agents are that they must be cooperative, competitive and proactive, have the ability to negotiate and perform in dynamic environments. To achieve this they must have the necessary communication methods (languages) and protocols. In addition, MAS are able decide whether to act or not based on the environment they are in and if the action is convenient or not.

An agent perceives its environment through sensors and operates through actuators to complete tasks autonomously. Agents have to be able to act and adapt to different types of contrasting environments, in some the agent has control over the conditions of the environment and in others it does not. These agents may have different architectures that will define how the agent determines the next action.

Agents can have a wide variety of properties, but the fundamental properties they should have are: autonomy, reactivity, reasoning, learning and communication. Giving too many properties to agents can lead to different types of complications. Privacy, security and technical management aspects must be taken into account in the implementation of any type of agent.

First we have the reactive agent architectures, which are able to react quickly to local events without having a very complex reasoning, they are simple and computationally inexpensive. Yet their interactions together can generate complex behaviors. On the other hand, we have deliberative agent architectures, which contain a model of the world they are in and therefore can make plans to make more precise decisions, but this entails a higher computational cost and greater complexity. Finally, we have hybrid architectures that mix qualities of these 2 systems.

It is possible to classify agents according to their properties, although it is usually easier to do so by the purpose they serve. There are 5 types of agents: Collaborative, Interface, Information, Facilitators, and Translator or wrapper agents. Other kinds of agents can also be designed depending on the task to be performed.

Communication is fundamental in multi-agent systems, in some cases it is important to have a way to resolve conflicts, a system for sharing tasks, sharing results and requesting help. Having communicative agents is important to achieve good coordination. They can be arranged in different categories such as the type of medium by which the message is sent or based on the intention of the message. There are different forms of communication, for example direct messages, via a blackboard, via a directory or a third party agent (facilitator). Communication has standards that will allow different agents on different machines to communicate. There are also protocols defined for the most common messages, such as asking a question, reporting and requesting help.

Coordination is when an agent reasons about its actions and those of other agents to find a coherent way to act together. Coordination is needed when actions among agents are dependent on each other and influence performance. It involves deciding what should be done, when it should be done and with whom it should be communicated, this generates a high computational cost. Coordination tasks must take into account the requirements and abilities of each agent to cooperate. Usually, agents are divided into independent and cooperative. Independent agents pursues its own interests, which can generate complications, while cooperative agents seek to fulfill general interests. Cooperative agents can be divided into explicit and implicit, where explicit agents send intentional signals to communicate and implicit agents observe and react to the behaviors of other agents.

The main objective of this work is to put into practice learned concepts about agent technology using Dedale. The exercise consists of the simulation of a treasure hunt scenario. The goal is to optimize the exploration and collection of treasures. This includes implementing different types of agents and creating coordination, communication, and cooperation protocols.

The world in which the agents exist is a map built as an undirected graph, where the agents move to achieve the goal. There are some constraints to take into account. Some sub-tasks need to be completed to achieve the final goal, and different types of agents specialize in certain tasks. An important consideration is the time and number of steps needed to complete the task. To do so, we have to think about the coordination mechanisms used to optimize the task and solve the problem efficiently.

In this first delivery, we present the proposed design of our solution plan. We analyze the problem, establish the characteristics of the multi-agent system, and select the types, functions, and properties of the agents. With these, we are capable of defining the architecture of our system in an informed way.

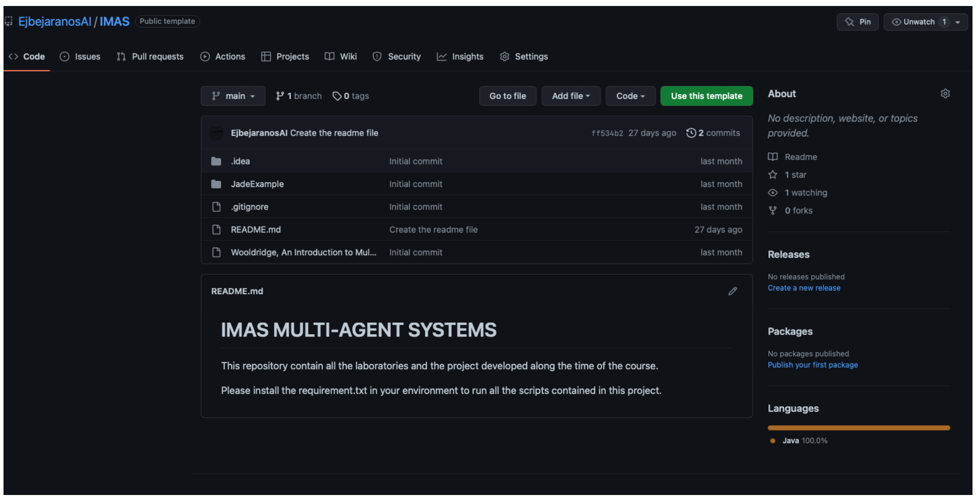
2 Methodology description

A well-defined plan is the cornerstone of a successful project, and for this project, a detailed plan has been defined, divided into different parts among the team members and implementing agile methodologies to optimize the results. Likewise, GitHub and its environment were used as complementary tools, to the version control and the manage times, tasks and processes.

For this project, the process was structured in the following parts:

- **Tools and software:**

  GitHub: As previously mentioned, some tools are implemented to facilitate project management, one of them was GitHub, which was used to control, share and manage code versions, which can be found in the following link: https://github.com/ejbejaranosAI/IMAS. Also, it was used to register, manage and control the different tasks of the project, likewise, a milestone was designed to plan the development of these tasks and be able to meet the stipulated delivery date. As shown in Figure 1. The project was divided into 3 parts, first, where was allocated to think about some problem that can be solved by complex networks and investigate possible solutions. The second was to understand and create a database that can simulate a scenario where it can be applied. The last part was to design and develop some recommendation system based on complex networks that can solve the basic requirements.



No description, website, or topics provided.

EjbejaranosAI Create the readme file ff534b2 27 days ago 2 commits

.idea	Initial commit	last month
JadeExample	Initial commit	last month
.gitignore	Initial commit	last month
README.md	Create the readme file	27 days ago
Wooldridge, An Introduction to Mul...	Initial commit	last month

No releases published

No packages published

README.md

IMAS MULTI-AGENT SYSTEMS

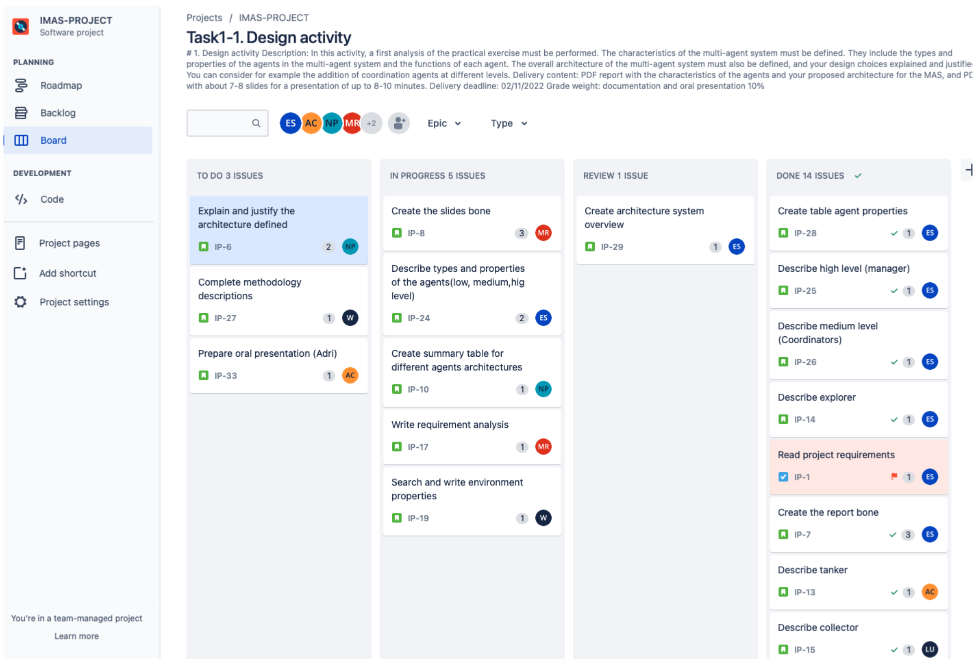
This repository contain all the laboratories and the project developed along the time of the course.

Please install the requirement.txt in your environment to run all the scripts contained in this project.

Figure 1 – GitHub repository

  Jira: It is used to facilitate the management of projects, and for this in particular it will be used as a tool that records each task and in the same way holds a participant responsible for carrying it out. In this way, the issue are distributed in the same proportion to all team members, managing to keep track of each of the tasks.

  In addition, a board with four columns was created,the first is 'things to do', where it is used to create problems for the future and is part of the planning. The second column is 'Doing', to reflect that the problem is being solved by some member of the team. The third column is 'Review', it is used to share with other members some previous development that required the supervision or the point of view of the team. Finally, the last column is 'Done', to refer to the issue or task being completed. Another important thing to mention is that, for each issue, a time is assigned to reflect the dedication in time, managing to estimate the work invested in hours.



Task1-1. Design activity

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 and the functions of each agent. The overall architecture of the multi-agent system must also be defined, and your design choices explained and justifie
You can consider for example the addition of coordination agents at different levels. Delivery content: PDF report with the characteristics of the agents and your proposed architecture for the MAS, and PD
with about 7-8 slides for a presentation of up to 8-10 minutes. Delivery deadline: 02/11/2022 Grade weight: documentation and oral presentation 10%

TO DO 3 ISSUES
- Explain and justify the architecture defined — IP-6 — 2 — NP
- Complete methodology descriptions — IP-27 — 1 — W
- Prepare oral presentation (Adri) — IP-33 — 1 — AC

IN PROGRESS 5 ISSUES
- Create the slides bone — IP-8 — 3 — MR
- Describe types and properties of the agents(low, medium,hig level) — IP-24 — 2 — ES
- Create summary table for different agents architectures — IP-10 — 1 — NP
- Write requirement analysis — IP-17 — 1 — MR
- Search and write environment properties — IP-19 — 1 — W

REVIEW 1 ISSUE
- Create architecture system overview — IP-29 — 1 — ES

DONE 14 ISSUES ✓
- Create table agent properties — IP-28 ✓ 1 — ES
- Describe high level (manager) — IP-25 ✓ 1 — ES
- Describe medium level (Coordinators) — IP-26 ✓ 1 — ES
- Describe explorer — IP-14 ✓ 1 — ES
- Read project requirements — IP-1 1 — ES
- Create the report bone — IP-7 ✓ 3 — ES
- Describe tanker — IP-13 ✓ 1 — AC
- Describe collector — IP-15 ✓ 1 — LU

Figure 2 – Jira board(Project management)

Java: Java is a high-level, general purpose programming language, one of the great benefits and strength of java is its portability, since it is designed to have very few environment dependencies as possible. We choose Java because when working on multi agent system, you're inherently working with different dependencies and configurations, and this is where Java excels because it be run directly. Java, in combination of domain-specific frameworks, can be an extremely power tool.

JADE: JADE (Java Agent Development Framework), is domain specific framework for Java, which enables you to develop multi agent systems. It's fully written in Java, which means it inherits Java portability. One of the main strengths of JADE is how you can switch up an agent between machines with very ease. Apart from this machine/agent abstraction, JADE has provides a set of GUI tools to remotely control agents, and peer to peer agent communication. JADE is compliant with the FIPA standard, which means it can produce production ready code that works with a variety of machines in different environments [7].

Dedale: The Dédale platform is a software tool that allows users to design, test, and run simulations of decentralized multi-agent systems. It provides a simulation environment and a set of tools for designing, testing, and running simulations of decentralized multi-agent systems. it is designed to be a realistic simulation tool, and allows for the creation of complex and adaptive multi-agent systems that can be used to study various coordination and decision-making behaviors[8].

- **Problem Statement**: In order to solve the problem posed by means of a multi-agent system, it is necessary to carry out a first analysis of our intended scenario. To do this, and facilitate the methodology in the project, it is necessary to answer the following questions:
  - What are our environmental properties?
  - What architecture did we choose for every type of Agent?
  - What are our agent properties?
  - What are our agent types?
  - How will those agents react between them? And with the environment?
- **Coordination:**

  At the time of starting the project, it was decided to carry out a division of labor, according to the main components of the project. That is why it was important to coordinate and divide the different tasks, that is why some strategies used in agile methodologies were adapted and with that roles were assigned to manage the project in the best way, that is why Natalia was assigned as scrum master and Lucia as product owner, and the rest of team as developers.

  However, it must be taken into account that as the project developed, the intervention of the different members in other tasks was necessary to understand, facilitate and adapt the new components in the system, so in the end an average of 20 hours per team member for the design part, in the following distribution:

- **Decisions:** As a fundamental piece for the development of the project, constant communication was necessary to be able to direct this project from scratch. One of the first decisions in the team was about how to approach the architecture and the types of agents, since we are not experts in that subject, so it was necessary to study that part for each member and then discuss them in a meeting. An initial configuration taking into account the lessons of the lectures, the decision was made to consult with the teacher and then iterate our idea and improve it, with the intention of better adapting it to the project.

  As a second important part in decision making, there was a discussion about how to present the ideas and how to distribute the tasks, since being a number of 6 people had to be coordinated correctly so as not to overload any member and that in the same way, all The tasks were completed satisfactorily and in the projected time. For this, the contribution of each member was essential to reach a consensus between ideas and activities.

- **Analysis:** After deciding how it is going to be the coordination and the decision making process in the group, we put our focus on the analysis of the problem, right after reviewing the theory from the lectures given in class. The items that we have analyzed in this project are the following:

  - Environment Properties
  - Agent Architectures
  - Agent Properties
  - Agent Definition

  The result of the analysis of these topics is the overall definition of the solution that we propose for this first delivery. This process is explained thoroughly in the following sections.

3 Requirements Analysis

The primary goal of this assignment is to demonstrate utilizing MAS software tools principles learnt in the theoretical lectures in a practical exercise. The issue presents a scenario of a treasure hunt where various types of agents must work together to gather the available treasures as quickly as possible.

The challenge must be broken down into multiple steps in order to accomplish the main goal. The first one is on system design, with a focus on the MAS's overall architecture. To finish this step, it's critical to develop the sub-tasks depicted in Figure 3.

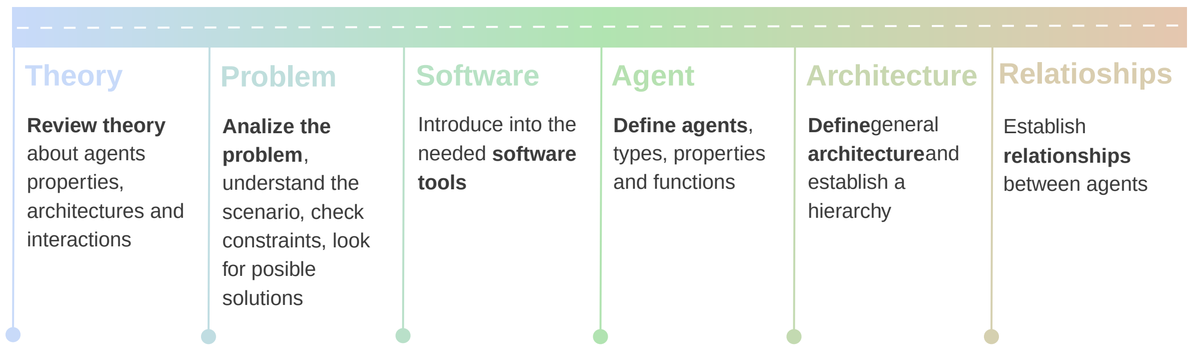

Theory
Review theory about agents properties, architectures and interactions
Problem
Analize the problem, understand the scenario, check constraints, look for posible solutions
Software
Introduce into the needed software tools
Agent
Define agents, types, properties and functions
Architecture
Definegeneral architectureand establish a hierarchy
Relatioships
Establish relationships between agents

Figure 3 – Stages to complete the system design

4 Multi Agent system analysis (Proposed Solution Design)

4.1 Environment Properties

Our environment is a graph that contains at least 500 nodes, the nodes represent unique places on the map. Thus our environment has the following properties:

- **Accessible**: It is possible in our case to gather complete and accurate and up-to-date information about the environment, and that makes our environment accessible. The agents do not need to create models of the world in their memories, because they can get any needed information from the environment at any time.
- **Deterministic**: Any action of our agents leads to the intended, expected and a guaranteed effect, there is no room for uncertainty, it can be determined by the current state or by the agent.
- **Discrete**: If the agent just has a certain set of possible actions that it can do in the moment, then the environment is discrete. Otherwise, when the agent has theoretically an infinite number of options, the environment is continuous. Our environment has a finite number of possible actions and that is what makes it discrete.
- **Episodic**: The agent's state in one episode has no impact on its state in another one. The agent can decide which action to perform based on its current episode, those episodes are independent of each other.
- **Static**: A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent. Static environments are the simplest since they remain constant and the environment doesn't expand or shrink.

4.2 Agent Architectures

Before starting with the configuration of the agents in the system, it is important to define the type of agent that you plan to use to solve the task in the existing world. That is why it is important to define interconnection, interoperability and cooperation with other agents in terms of terms.

Typically agents are classified by their purpose. The common 5 types of agents are:

- **Collaborative agents**, communicate, cooperate and negotiate with other agents.
- **Interface agents**, focus on automating tasks and carry out learning to adapt to the user.
- **Information agents**, manage access to different sources, extract information and present it to other agents or to the user in a personalized way.
- **Facilitators**, manage the internal connections of the system and the collaboration between agents.
- **Translator and wrapper agents**, focus on the communication between systems, even if they are agentified or not.

In addition, other types of agents can be created depending on the task to be solved.

In our case and due to the nature of the Manager, and as it is observed in the table 1, it can be affirmed that the type of agent for this is as interface agent, since it emphasizes communication and cooperation with the external user.

For the coordinators, it was proposed to use a facilitator agent type configuration, because, with that each agent cant talk directly to any other agent, also, have a facility to manage the connection between then.

Table 1 – Agent types in the treasure hunt problem

Type	General Manager	Coordinators	Explorers	Collectors	Tanker
Collaborative			✓	✓	✓
Interface	✓				
Facilitator		✓			

At the end, for the explorers, collectors and tankers it was thinking to use a collaborative configuration, because it has a great ability to negotiate and solve problems, also to coordinate .

Table 2 – Agent architecture in the treasure hunt problem

Architecture	General Manager	Coordinators	Explorers	Collectors	Tanker
Reactive			✓	✓	✓
Deliberative	✓				
Hybrid		✓			

- **Reactive**: are able to react quickly to local events without having a very complex reasoning, they are simple and computationally inexpensive [1].
- **Deliberative**: contain the model of the world and therefore can make plans to make more precise decisions, but this entails a higher computational cost and greater complexity.
- **Hybrid**: Mix qualities of reactive and deliberative systems. Usually the reactive part has dominance over the deliberative part and they are introduced in the form of decision layers. Although there is no general methodology to make them and they are not supported by formal theories.

4.3 Agent Properties

Regarding agent properties, we define our agents as shown in the following figure, rating them with different capabilities depending on the type of the agent. As it can be seen, higher level agents such as the manager generally have more complex properties than lower level agents [2].

Table 3 – Agent properties in the treasure hunt problem

Properties	Manager	Coordinators	Explorers	Collectors	Tanker
Autonomy	High	High	Medium	Medium	Low
Rationality	Yes	Yes	Yes	Yes	Yes
Reactivity	Low	Medium	High	High	High
Mobility	No	Yes	Yes	Yes	Yes
Learning	Yes	Yes	Yes	Low	Low
Social ability	High	High	High	Medium	Low
Reasoning capabilities	High	High	Medium	Low	Low

4.4 Agents

In order to achieve the goals of the treasure hunt problem, we will implement different types of agents for each of 3 levels. That is why it is important to create a hybrid system, which attempt to get the better of two sides (Classical and alternative approaches). With this, 3 different levels it is proposed for this architecture with the intention of solve the problem dividing into smaller sub-problems (synthesis solution), this is typically a hierarchical process, they are described as follows.

4.4.1 High level (Manager)

At this point of hierarchical top, it is proposed a manager that have the aim to control the coordinators and manage the communication between all of them, and also maintain the control of the behaviour of all the elements on charge and verify that the rules of the actors are attaching to its goal. This kind of agent needs a deliberative architecture to maintain the focus on long term plan actions, in other words, as head it is necessary to stay centered on a set of goals. Collects results, eliminates completed tasks, prioritize tasks, manage global goal, manage statistics at each step time.

4.4.2 Medium level (Coordinators)

As coordination in this system, it is necessary to manage the dependencies between the types of tasks (explore, collect and tank) and their agents. On the other hand, for coordination to be positive, it is necessary to maintain explicit communication between agents and low-level actors. level.

Regarding the architecture of these coordinators, it is proposed to have a hybrid architecture that combines the reactive and deliberative side in the same point. With this, it will be possible to maintain rapid responses to changes that low-level agents need, while trying to maintain focus. of the task assigned by the manager.

- Tanker Coordinator: keeps track of the position of tankers, receive petitions from collector coordinator, communicate with tankers.
- Explorer Coordinator: keeps track of the position of explorers and communicate with them, receive updates on the map and share it with other managers, receive locations of treasurer, send petitions to collector coordinator.
- Collector Coordinator: keeps track of the position of collectors, send petitions to tanker coordinator, communicate with collectors, send collectors to explorer's/treasure's positions.

4.4.3 Low level (Actors)

At the low level, it is proposed that the architecture of the agents should be focused on quick reactions, with the ability to react to changes found in the environment. Another reason for choosing this type of architecture is that, as is known, these agents will represent the largest number of actors, and as a good practice, it is decided to keep these agents as simple and computationally economical as possible. In addition, with this architecture we can provide robustness against possible failures.

- **Explorer**

  These actors aim to explore the environment that is proposed to them. In addition, they will have the ability to share their acquired knowledge of the map with the coordinator and other explorers to gradually build the map of the entire environment through random movements. As limitations on this actor, he cannot collect objects or treasures since those tasks are intended by another agent, even though he possessed the best strength and lock-picking skills.

- **Collector**

  The main goal of the Collector type agent is to collect the treasures spread in the map. It has a limited capacity backpack to carry the treasures, which can be either gold or diamonds, but just one type of them. The amount of treasures it can grab is set by the backpack capacity of the agent. Also, if a Tanker agent is in communication range, it can drop the treasure of its backpack to the Tanker. Finally, regarding the unlocking abilities, it has limited abilities to open safes.

- **Tanker**

  The tanker agent is the one mainly in charge of storing the treasures collected by the other agents. It is able to move around through the map and communicate, but it has no strength neither lock picking skills. So all the treasures it will carry must be delivered by the collectors.

The tanker agent must then be able to receive treasures by other agents and update its backpack information. It must be able to move around the map. It will also be the most simple agent among the low level ones, since we won't make it find other agents. The main idea is that tankers move around the map and others go find them.

4.5 System Overview

In this section we describe how the whole system will work, considering the described agents and the environment they will operate on.

Given the environment and agents we described in previous sections, the way we unify them can be visualized in figures 4 and 5.

Figure 4 shows the hierarchy between different agents. Manager is the highest level one, in charge of designing a plan and orchestrating the rest of agents to achieve it. There is a helper, medium level layer between the manager and the problem's agents: coordinators. There is one for each low level agent type and will be in charge of communicating with other class coordinators and the manager itself. They will also give direct orders and receive updates from the lower level units of their respective classes. Finally, the low level agents are the explorers, collectors and tankers themselves. They will be in charge of executing the plan designed by higher level agents.

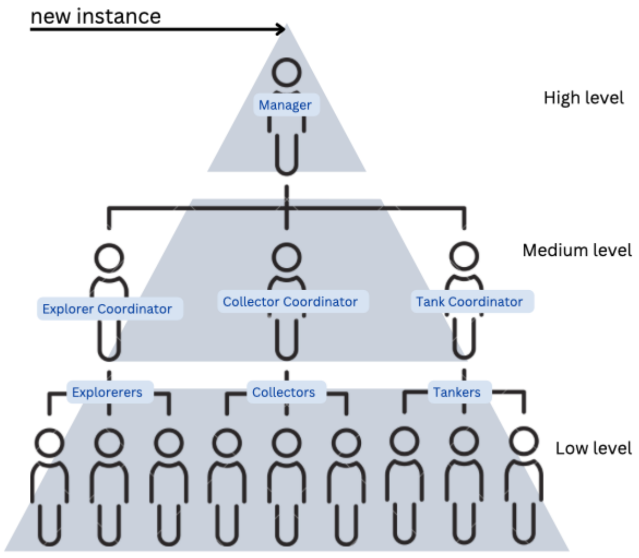


Figure 4 – System architecture overview

We can visualize the relationships between the entities of the problem in Figure 5. We see the same pyramid structure, where the manager is on top managing and receiving reports from coordinator level agents. Those act as intermediates between the plan of the manager and their respective agent units. In the base there are explorers which store and share the map they have discovered between other explorers, and they are also able to unlock treasures, but not carry them. Collectors on the other side are also able to unlock treasures, and they can also carry them with a limited space. To overcome limitations they are able to deliver treasure to the tankers. Tankers are just able to carry treasures.

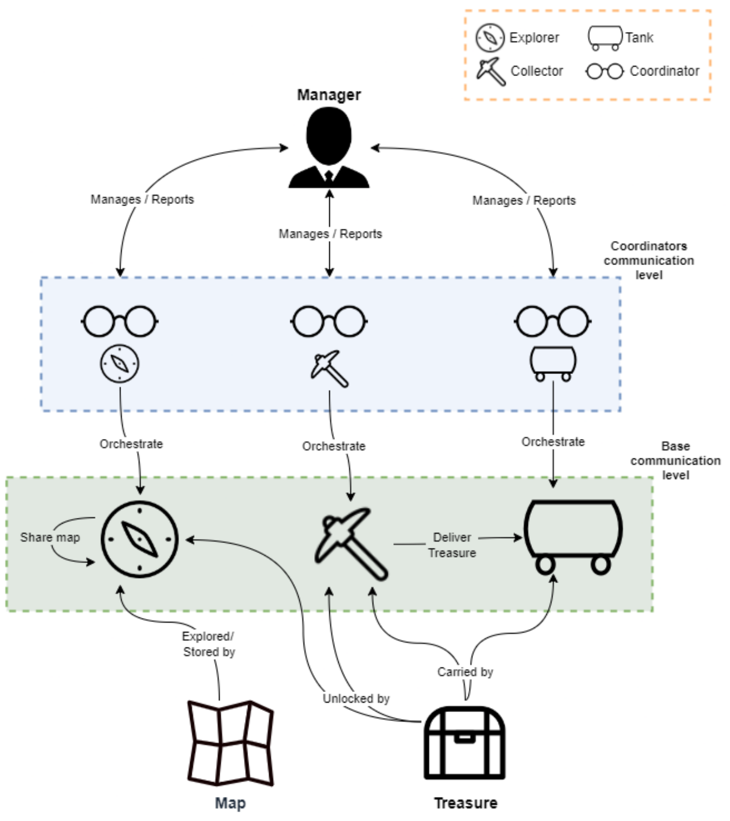


Figure 5 – Relationship between entities

4.6 Architecture justification

This structure was chosen due to the conditions and restrictions that the agents have at the moment of executing their tasks and achieving the final objective. It is necessary to have a certain level of coordination in order to have an overview of what is happening in the system. In addition, it is necessary to record the progress obtained at each moment to know the current status and to be able to plan the next action in the best way.

For this reason, we have decided to establish 3 levels of agents. The first is the lowest level, being the agents predefined in the problem: explorer agent, loader agent, and collector agent. They are low-level as they focus on field actions, such as treasure hunting, map exploration, and collecting.

Subsequently, we have a medium level which are coordinators per group. These coordinators help to make communication more fluid and to better manage resources. That is, they coordinate the agents used in each section of the map according to the progress made at each moment of time.

Finally, we have the highest level, the manager. This will help us to plan and mark the steps to follow in the future goals, and record the results and the progress made as the simulation progresses.

It should be noted that low-level agents are reactive since it is not necessary for them to have a very high level of planning, but to react at the moment and when they need to perform their tasks.

The mid-level agents are of the hybrid type, as we need them to act quickly but also be in coordination with the general manager and be able to plan the development of subtasks to complete the overall goal. Finally, we have chosen the general manager to be a deliberative type as it will be in charge of maintaining a fixed course to complete the task.

The relationships between the agents are carried out according to the level of hierarchy established. The low-level agents communicate with the middle-level agents and in turn, the middle-level agents communicate with each other and with the highest level.

5 Communication, Coordination and Negotiation

In this section analyzing the process of problem solving that agents will use to achieve a common goal. This process involves several steps like communication, voting, negotiation, and general coordination.

Section 5.1 is an overview of the communication categories and their classification and the configuration we chose to solve our problem.

Section 5.2 gives an overview of the different coordination tasks in this specific problem.

Section 5.3 analyzes the different possible negotiation mechanisms that could be chosen to solve the problem: Auctions, Voting ,Coalitions and Implicit communication, with a justification of why they were chosen.

Section 5.4 mainly describes the behavior of every agent in our system, a pseudo-code was written to better describe those behaviors.

In a distributed problem-solving environment, a problem goes via several steps:

- Task Decomposition: In this stage, a higher deduction agent (in our case, a general manager) would decompose the task into a set of smaller, more doable tasks between its coordinates.
- Task Allocation: once a general manager has passed said tasks to each specific niche agent, those agents would in themselves pass the actions into smaller agents, in a process called task allocation.
- Task Accomplishment: In this step, each agent would start doing their part and communicating the results up, from low level agents to coordinator, to the general manager, who can decide on the next steps.
- Result Synthesis: In this step, a general manager receives the result before deciding on how to go next in a process called result synthesis.

In our system, problem decomposition happens on the general manager and coordinator level.

Problems are decomposed following the capacity of agents and their skills, and relies heavily on cross communication.

Agents communicate between themselves via the general manager and coordinator petition system, low level agents rarely need to communicate if at all. A general manager is responsible for detecting and managing conflict that arises between coordinator, while those are responsible for their agents, via a voting and a bidding process.Results are always communicated upward to a general manager and are merged there.

5.1 Communication

As one of the most important things in the multi-agent system, is the relationship between them, and the ability to communicate information in order to cooperate and solve any problem. Optimal communication can increase performance by solving complex problems, reducing them to solving easier sub-problems by working together. On the other hand, a collective effort requires coherence, competence and coordination, and more so in distributed problems, where the coordination between many actions carried out by the agents must allow the sharing of information to coordinate everything.

5.1.1 Communication categories

Likewise, within communication there are some categories where they are classified in 3 manner mechanism:

- **Via the type of sendee-addressee link:** Here exist three possible configurations:

1. **Point to Point**
2. Broadcast
3. Mediated

In our case it is going to be used **point to point** configuration, because basically the agent can talk directly to another agent, in other words, a single receiver from a single sender, in a direct route.

Broadcast and mediated are not considered, because it is desired to maintain simplicity in the system, since for the broadcast it is necessary to locate the agents and this leads to implementing an internal directory, sometimes simple is more efficient.

The way it was discussed for configuration at all agent levels was:

1. **Low level:**
   - Explorers (share the map Explorer ←→ Explorer) ←→ Explorer's Coordinator. (Point to point)
   - Collectors ←→ Collector's Coordinator. (Point to point)
   - Tanker ←→ Tanker's Coordinator. (Point to point)
2. **Medium level:**
   - Mediated (coordinator notify others coordinators by a third party)
   - Mediated (Coordinators ←→ Facilitator ←→Low level agents)
3. **High level:**
   - Point to point (Manager ←→ Coordinators)

- **Via the nature of the medium** For this category, it is the medium of how the message it is propagated, in this case it is know that exist 3 kinds if natures, that are:

  1. **Direct routing**: Here, the message is sent directly to another agent without interception or fading. Also, another positive aspect of this configuration is that the nature of the information can be very varied. For this reason and as mentioned above, this configuration was selected, since it is part of a decentralized communication that can simplify the cooperation between agents.
  2. Signal propagation routing: The agents, in most of the cases of architecture reactive, sends signal whose intensity decreases according to distance.
  3. Public notice routing: In this configuration, a use of blackboard systems is implemented, to have a common station, where the agents place the information to be able to share it with others. The problem with this configuration and why we are not taking it, is because it represents a centralized structure where in turn it can present a bottleneck in the system, which represents a high risk since it can lead to a single point of failure.

- **The type of intention** On other side, in the types of speech acts it was determined to use:

  - **Inform** (For coordinators to report to manager - For low level agents to report to respective coordinators)
  - **Query** (So that the manager and coordinators can ask about the current situation of their managers)
  - **Answer** (So that agents can answer questions asked by other agents)
  - **Request** (So that coordinators can request to act the low level agents)

5.2 Coordination Techniques

Coordination between agents depends on their ability to cooperate. Cooperation in multi-agent systems can be divided into two groups according to Franklin cooperation hierarchy: independent/self-interested and cooperative/benevolent.

Agents which belong to the first group are assumed to act to further their own interests, and they will possibly do that at the expense of other agents. Besides, implementing this kind of cooperation may lead to conflict between the agents. Because of all that reasons it is obvious that self-interest cooperation is not a suitable solution for our problem.

Benevolent agents, on the other hand, consider that their best interest is the best interest of the compound of the agents, so cooperative distributed problem solving seems like a better fit for the treasure hunting task. On a deeper level, benevolent agents can be split into two different groups: agents which perform intentional sending and receiving of communicative signals (explicit) and agents who don't explicitly communicate via messages (implicit).

First, we will talk about explicit coordination, which can be divided into two disctint groups: deliberative agent coordination and negotiator agent coordination.

5.3 Negotiation

5.3.1 Negotiator agent coordination. Voting

Voting is a cooperation protocol in which different agents have to choose among a set of actions the best one according to their criteria. Then the most voted action is the one that will be chosen.

For this problem we will use voting primarily to choose the order of the treasures to collect. At a given point, the coordinators will know how many treasures have been found by the explorers and are yet to be collected. Then, the collector's coordinator can start a voting between the collectors agents to decide which treasure should be collected first. Each individual should vote considering which treasure is closer to them and if they are available or not, giving priority to the most dense and unoccupied areas first. Then, in order to choose who will collect it we will be using an auction protocol, which is described in the following subsection.

For this to work we first need to let all the agents that a voting is taking place and which are the options to choose. Then each one of them has to take a decision and forward it to the coordinator, who will be in charge of gathering the results, get the final decision and proceed with the following action accordingly.

We will be using a **Plurality protocol** where each agent can give 1 vote to 1 of the alternatives. The alternative with the highest number of votes wins. We don't consider the problems of this protocol regarding the possibility of minoritarian groups coaliting to win the voting, since each agent will vote truthfully only regarding its position.

The main issue we could have is a tie between two or more options. To solve this, the coordinator could either choose randomly (since any of the chosen options will be valid and will have to be collected at some point), or ask another coordinator for a second opinion based on other types of agents locations. The manager could also help to solve these conflicts if the coordinators cannot solve them by themselves.

5.3.2 Negotiator agent coordination. Auctions

As a negotiation process, auctions help to achieve goals within the environment of a multi-agent system. Specifically, there are cases where resource constraints require efficiency in the execution of a given task. Auctions are also useful to reach an agreement in situations where agents are in conflict and a competitive environment is generated. In order to participate in negotiation protocols, agents must follow a set of rules. The end criterion of the auction is reached until a legitimate bid is accepted. The number of units determines the number of offered instances.The number of attributes to be taken into account in the bids of each agent depends on the purpose of the auction.

For the treasure hunt problem, we will use a **first-price sealed-bid multi-attribute auction** to choose which agent collector is the best option to help a explorer who found a treasure, in order to open the chest and collect the treasure. This same strategy for auctions can be followed to select which tanker should go to store a collected treasure.

For the case when a collector is needed by an explorer to grab a treasure, the explorer would notify its coordinator the found treasure with its current location and the type of treasure. If it has been capable to open the chest by its own, the explorer can continue with its walk. In case the explorer can not open by itself the treasure and the abilities of other agents are needed, the explorer will wait in the treasure's node until a collector is in a proper range of distance to combine abilities and open it. Then explorer's coordinator will communicate the collector's coordinator and tanker's coordinator its needs to collect the found treasure. In this case the collectors coordinator will call for an auction among all the collectors. Collectors will send their location and its capacity to carry both types of treasure. With this the collector's coordinator will chose the best bid and notify to the winner agent the location of the treasure to be collected. Meanwhile the tankers coordinator will do another auction between the tankers to decide which will follow the collector and carry the gathered treasure.

The rules for participating in an auction must meet the following criteria. As an admission rule, the type of agent that can participate in an auction must be the same type of agent as the coordinator who call for the auction. As an interaction rule, agents must send their proposals directly to the coordinator. As a validation rule the agent must be without a major occupation at the time of bidding (e.g. collecting another treasure). The outcome of the auction is chosen by the coordinator who convenes the auction, based on the best bid received.

Including auctions in our system is intended to give flexibility to reduce the time needed to make a decision in certain scenarios. In our case each auction involves only one instance at a time. The buy bid is the actual attributes of an agent and the sell bid is the optimal attributes to complete a task. This in general simplifies the task of assigning activities to specific agents. To summarize, the auctions proposed in our system follow these criteria:

Auction process:

- Coordinator agent calls an auction
- Agents of the same type and meeting the requirements submit their current status attributes
- The coordinator receives the proposals and chooses the most optimal one
- The coordinator notifies the winning agent of the auction with the location of the treasure

Auction properties:

- Single good
- Multi-attribute
- Sealed-bid
- Ascending
- Single unit

5.3.3 Deliberative agent coordination. Coalitions

Deliberative agents have inference and planning capabilities; in order to solve collectively a given problem, deliberative agents make use of a distributed organization mechanism based on information exchange. In class, we have seen two methods: PGP and coalition formation. Coalitions are temporary collections of individuals working together for the purpose of archiving a task. These are the cases where the task can not be performed by a single agent or it can be performed more efficiently by several agents working together. Nevertheless, we won't be implementing a deliberative coordination coalition mechanism since the decision making process will be performed by higher level agents which will decide the best action to take in each moment. For this reason, the task of the lower level agents will be to communicate their state to their respective coordinator and act upon their decisions.

5.3.4 Implicit communication

In implicit coordination, although agents do not communicate with each other, the designer of the system intends to provoke the emergence of the socially intelligent problem solving activities. Some of the motivations for using implicit coordination are speed (when it takes too long to communicate a message) or security (when you want to keep your plans as a secret); other reasons of using this kind of coordination could be that the task is too complex to be executed by simple agents or that there is a lack of communication channels.

One of the forms of implicit cooperation is the indirect cooperation through the effects on the environment of the actions of each agent. Here, agents do not communicate between each other, in fact, agents modify the environment and the others react to the change and modify their actions accordingly. This type of cooperation mechanism won't be implemented in our work, since communication will be done via explicit messaging between agents. Other form of implicit communication is to use reasoning mechanisms at individual or societal level. Regarding the latter, methods that try to impose some kind of organization in the multi-agent system can be conceived, such as product hierarchy, functional hierarchy, a mix of both or flat structure. The architecture which most resembles our proposed solution is the functional hierarchy, where there is a global coordinator on top, one coordinator for each of the three agent types, and finally under the administration of those managers, the agents themselves. These low level agents do not communicate between each other directly, they follow the rules imposed by the organizational structure that has been designed.

5.4 Agent behaviours

In this section we collect the different pseudocodes that describe the behaviour of each agent during the treasure hunt simulation.

5.4.1 Collector

Algorithm 1: Pseudocode - Collector behaviour

```
while True: do
    Move to a random position in the map
    Search treasure
    if Treasure = True and (canOpen or !isLocked) then
        Open Treasure
        if canCarryTreasure then
            startMission(TreasureFound)
        end
    end
    if Vote Petition = True then
        Vote
        Send vote to Explorer coordinator
    end
    if Auction Petition = True then
        Bid
        Send bid to Explorer coordinator
    end
    if Auction winner = True then
        startMission(TreasureIndicated)
    end
end
```

Algorithm 2: Pseudocode - Collector startMission()

```
Data: $Node(Treasure)$
Go to Treasure
if isLocked and !canOpen then
    Mission Failed
    return
else
    openTreasure
    while !TreasureEmpty do
        pickTreasure
        findTanker
        leaveTreasure
        Go to Treasure
    end
    return
end
```

5.4.2 Tanker

Algorithm 3: Pseudocode - Tanker behaviours

```
while mission: do
    Move to a random position in the map
    Wait for Auction call
    if Auction = True then
        Send status to coordinator
    end
    Wait for Auction result
    if Auction = winner agent then
        Search collector
        Receive treasure from collector
        Updated backpack information
        Update map information
        Move to another random position in the map
    end
end
```

5.4.3 Explorer

Algorithm 4: Pseudocode - Explorer behaviours

```
Data: $Graphnodes(Map)$
Result: $TreasuresPosition$
while mission: do
    Move to a random position in the map
    Search treasure
    if Treasure = True then
        Get position of the treasure
        Inform the coordinator agent about the position of the treasure
    end
    Updated map
    Move to another random position in the map
end
```

5.4.4 Collectors coordinator

Algorithm 5: Pseudocode - Collectors coordinator behaviours

```
while mission: do
    Request information about the position of the collector agents
    Update map with the position of collectors
    Wait for petition from Explorer coordinator
    if Petition = True and Next Treasure = False then
        Start voting process between agents
        if All agents voted = True then
            Next Treasure = selected
        end
        Send next treasure location to collectors
        Send information to General Manager
    end
    if Next Treasure = selected then
        Start auction protocol
        Wait for bidding from agents
        if All agents bid = True then
            Select best bidder from list
            Send Collector to next treasure location
            Next Treasure = False
            Send information for General manager
        end
    end
    if Collector capacity full = True then
        Petition Tanker coordinator about nearest empty tanker
        Send Collector agent to location
        Collect statics information
        Send information for General manager
    end
end
```

5.4.5 Tankers coordinator

Algorithm 6: Pseudocode - Tanker Coordinator behaviour

```
while mission: do
    Wait for petition from Collector coordinator to auction
    if Auction = True then
        Send auction petition to Tanker agents
    end
    Wait for results from auction
    if Results = True then
        Receive auction results and select Tanker winner
        Send winner message to Tanker
    end
    Send information to General manager
end
```

5.4.6 Explorer coordinator

Algorithm 7: Pseudocode - Explorer coordinator behaviours

```
while mission: do
    Request information about the position of the explorers agents
    Updated map with the position of the explorers
    if Treasure is founded by some explorer = True then
        Receive inform from the explorer
        Collect position of the treasure
        Update map
        Collect statics information
        Send information to General manager
        Share information map with all the agents
    end
end
```

5.4.7 Manager

Algorithm 8: Pseudocode - General Manager behaviours

```
while mission: do
    Receive information on the state of the environment
    Manage the general statistics(collectedTreasures, stepsNum, availableResources)
    if Discovered treasures to be collected yet $\geq 3$ then
        Call for a vote between coordinators
        Collect the vote of each coordinator
        Report the results of the vote
        if Tied vote then
            Vote to break the tie
        end
    end
    Update the statistics report
    if collectedTreasures == 10 then
        Present the general statistics of the simulation
    end
end
```

6 Implementation

Up until now, we have designed several aspects of our multi-agent system for the treasure hunt problem. Among the characteristics that have been designed we find the agent architecture, agent properties, the communication methodology (coordination and negotiation strategies) and finally the expected behaviour of each kind of agent. Theoretically, we consider our designed well defined and closed. Nevertheless, once we have started implementing the design of our IMAS, we have realized that the architecture is unnecessarily complex to solve the task efficiently. As a result, we decided to simplify the design in order to make it feasible to implement and coordinate agent actions dexterously. This involved streamlining the interactions between the agents and reducing the number of agents that were involved in the system, among other things. With this updates in the design, we were able to successfully implement the system and get it up and running.

6.1 Rethinking the architecture

One of the updates we did is getting rid of coordinators and manager. Coordinators and manager are typically used in IMASs to oversee and coordinate the activities of the other agents in the system. However, these types of agents can add unnecessary complexity to the system, especially if they are not strictly necessary for the problem at hand. According to our tests, implementing communication steps between low level agents with coordinators, and between coordinators and a general manager, generates most of the times (for our proposed solution) additional and unnecessary messages management. This implies that the system gets overload with messages that need to be attended and it slows down the program execution. Additionally, we notice that the restriction of the 3 nodes communication range cause the coordinators get in the way of the other agents, blocking routes and delaying the completion of the problem. In our case, the explorer, collector, and tanker agents are able to communicate and coordinate their activities effectively without the need for additional coordinating agents.

Although we did not use the chosen cooperation protocols as stated in the initial proposed plan, we implemented coordination protocols to manage specific situations that arise during the simulation experiments which uses some mechanisms seen during the class that behaves as negotiation and auctions. Auctions and biddings are often used in IMASs to allocate resources or distribute tasks among agents. However, the use of these mechanisms requires a lot of continuous communication between the agents. This does not fit well with the nature of our system due to the agents have range limitations in the communication and they have to be moving most of the time. Forcing these behaviours could became unnecessary for the treasure hunt problem we are trying to solve. Additionally, depending on the type of problem, other communication or negotiation techniques such as social laws and agreements can be used to facilitate agents coordination.

The updated architecture has a horizontal format as can be seen in the figure 6. This allows for a more efficient communication flow and less time consuming decision making. However, due to the tools provided by dedale this does not affect the effectiveness of the decisions made by the agents for the completion of subtasks. In this new architecture, the agents with the greatest coordination and problem-solving capacity are the collectors supported by the explorers. The tankers try to cover large areas of the map so that they can be easily found by the collectors who need to deliver treasures.

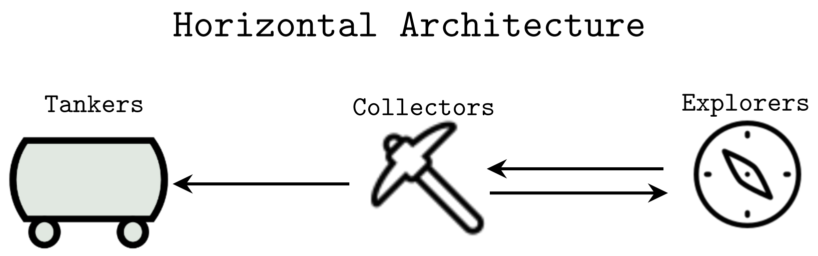


Figure 6 – New horizontal architecture

6.2 Modifications in the agents

6.2.1 Explorers

Generating knowledge of the map

The main function of the explorer agents is to generate knowledge about the map so that it can be shared with other agents and they also have a representation of the world they are exploring. Thus, during the exploration, explorer agents keep track of both the nodes they have visited, in order to generate the graph representation of the map, and also keep track of the treasures that they have found along the way and they're exact location in the map.

During the exploration phase of the map, explorer agents may meet other explorer agents which have also been exploring the map. When two explorer agents meet in the communication range, they have been given the ability to share their knowledge, meaning that they share the maps that they have been constructing. In that way, it is not necessary that all the explorer move through every node of the map in order to obtain the full map representation. Besides, there is also another benefit of sharing the maps between each other, which is that they finish earlier the exploration phase.

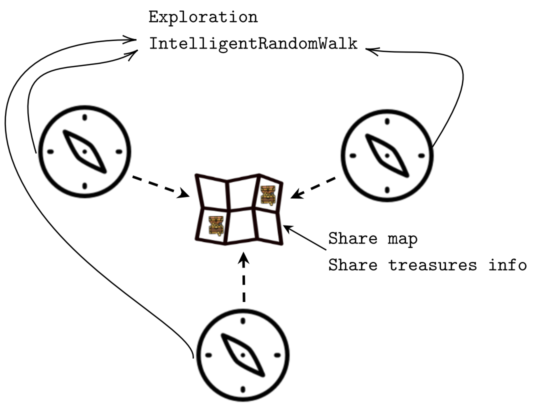


Figure 7 – Explorers Sharing-Map Behaviour

Explorer's communication with collector agents

Eventually, explorer agents will finish exploring the map and will have its full representation. In the first implementation of these kind of agents, once they had finished exploring the map they remained frozen. Nevertheless, this generated some issues, for example the blocking of paths of other agents who where on their mission to collect or to receive treasures. Because of these, the implementation was changed. Now, the explorer agents, once they have finished exploring the map, they start a random walk along the nodes of the map. During this walk, explorer agents will meet collector agents and they will communicate the following way. Collector agents will send a message to the explorer asking for the location of a treasure of the kind it can collect. Then, the explorer agent will access the information it has gathered about the treasures and the nodes and it will provide the collector agent with the shortest path to the treasure it can collect. Finally, the random walk will initiate again until it meets another collector along it's path and exchanges information about another treasure.

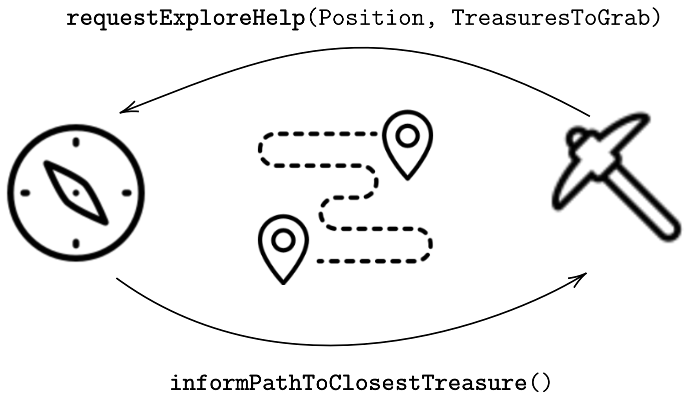

requestExploreHelp(Position, TreasuresToGrab)
informPathToClosestTreasure()

Figure 8 – Explorers Interaction with Collectors

Ability to open the treasures

According to the initial conditions of the problem, the explorers have high levels of lockpicking and strenght expertise which is necessary to open some of the treasures on the map. Because of this during both their exploration stage and their random walk stage, every time they encounter a treasure they try to open it and report whether they have been able to do so or not.

Algorithm 9: Pseudocode - Explorer Agent behaviours

```
while mission: do
    Walk along the map avoiding visited nodes
    if Map exploration = finished then
        Initialize random walk
        if Explorer meets Collector then
            Provide with shortest path to treasure
        end
        if Explorer meets Tanker then
            Provide with location near treasure
        end
    end
    if Explorer gets stuck then
        Take a 10 step random walk
        Continue looking for unexplored nodes
    end
end
```

6.2.2 Collectors

Main behaviour

On the other hand, the collector, will perform several actions in every tick of the clock:

- **Treasure hunting.** The agent will check if in the current node they are in there is a treasure of the type it can collect, if it is and if it has the necessary ability to collect it, it will unlock the treasure and put it in its backpack.
- **Treasure delivering.** Besides, in every node, the agent will check if there is a tanker agent in its communication range and if there is, it will try to pass it the treasures.

Improvement in random movement

The movement of collector agents will start at random. The pre-defined random movement has some issues though, as the agent will find itself moving in every tick forward and backward an not advancing a lot in the path. Thus, we have implemented a function which optimizes the movement of the agent. According to this function, the agent will keep track of the last nodes that it has visited and will try not visit them again, unless they get stuck at a node and have to go backwards or to other nodes they have passed through before.

Solving conflicts between paths

When two agents are following the same path in opposite directions or when they intersect in the same node, it can happen that both agents collide in the same node and are unable to solve the conflict and keep moving. Besides, collectors and tankers do not have the information about the map, so they do not have the required knowledge to solve the problem.

The proposed solution is the following: determine the priority of each agent in relation to their mission and the agent with most priority will be the one following the its predefined path. But how to determine which agent has more priority? In our implementation, the two agents will share their paths and the agent who has less nodes to finish their path it will be the one with most priority. Thus the agent with no priority will step back the necessary steps in order to let the agent with most priority to continue their path until they reach the final node or they encounter another agent and the priority measuring process will begin again.

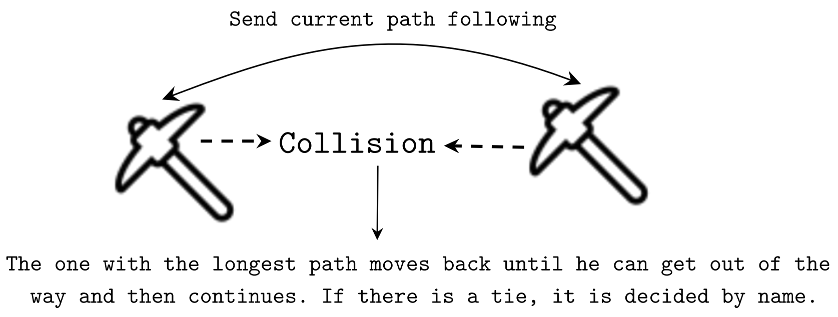

Send current path following
Collision
The one with the longest path moves back until he can get out of the way and then continues. If there is a tie, it is decided by name.

Figure 9 – Collectors avoiding collision behaviour

Communication with explorer agents

The explorers go through the map and write down in a list the treasures they have found. Within this information they keep both the type of treasure and the node where it is located.

When the collector meets the explorer, the collector asks him for the shortest way to the treasure he can collect, then the explorer gives him a path and he follows it. The next step is to find a tanker to which to pass the treasures found. This tanker, as will be explained later, will remain in a fixed node waiting for the collectors. In case there is no tanker fixed to that node, the collector will have to look for it in a random move.

Communication for sharing treasure information

The collector agents keep a list of the treasure they have found, in this list they add two different types of information: the amount of treasure and its type. Every time the collector meets an agent while he is walking the map, it shares with that agent by means of a message its list with the information about the treasures found. Similarly, the agent it meets does the same action in a reciprocal way. It sends a message with the information it has gathered. In this way both agents complete their lists with the new information about the treasures that they have not seen but that are scattered around the map.

However, it is possible that by the time both agents share the information in their notebooks, another agent has already passed by those treasures and they have already been collected in whole or in part. This is one of the benefits of sharing the notebooks of each agent, if one of them had noted that x treasure had x amount and it turns out that, according to the information shared by the other agent, there is less amount of that treasure, the agent will update the new information in his notebook.

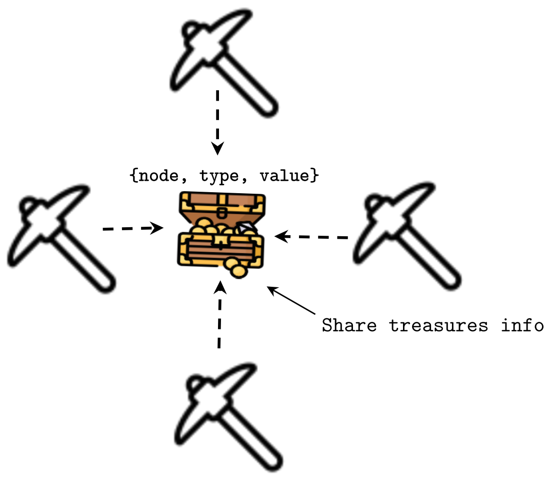


Figure 10 – Collectors Sharing Treasure Information

Algorithm 10: Pseudocode - Collector Agent behaviours

```
while mission: do
    Random walk along the map
    if Collector meets Explorer then
        Ask for shortest path to treasure
        Move to treasure node
        Collect treasure
    end
    if Collector meets Tanker then
        Unpack treasure on tanker
    end
    if Agent collides with other agent then
        Exchange paths
        Compute priority
        if Not priority then
            Moving away until avoiding collision
        end
    end
end
```

6.2.3 Tankers

Initial behaviour

Tanker agent's main objective is to collect the treasures that other collector agents have found and they do not have the needed storage in their backpacks. Initially, when the simulation is started, all of the tanker agents move in a random way. This behaviour will remain that way until the explorer agents have finished exploring the map and generating knowledge about all the nodes and the edges.

Communication with explorer agents

Once the explorers have finished, they start a random walk. During this walk, at some point of the simulation, explorer and agent will match in the same communication range, here is when the communication between the two agents start, and it follows the next steps:

- Explorer agent sends a request to the tanker asking about its ID and its location in the map.
- Tanker answers explorer with the information it requires

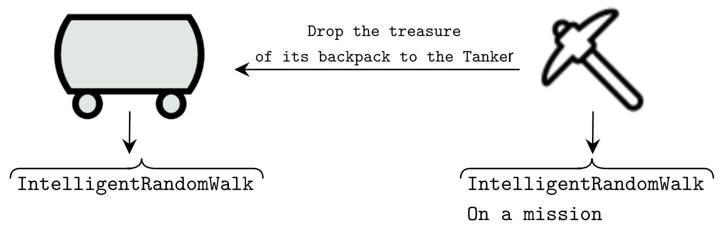


Figure 11 – Tankers Interaction

- Explorer accesses the information about the map and the location of the treasures and sends the tanker the path to the treasure it must locate nearby.

That way, when tankers go where explorers have sent them, they will wait for a collector to come and to receive the treasures it has gathered in the treasure location nearby.

Algorithm 11: Pseudocode - Tanker Agent behaviours

```
while mission: do
    Random walk along the map
    if Tanker meets Explorer then
        Ask for instructions on where to move
        Move to destination node
        Wait to receive treasures from Collector
        Remain in that position until end of simulation
    end
end
```

6.3 Summary of the new implementation

In the updated implementation, the system has been designed to be distributed, allowing for parallel processing and increased scalability. Explorer agents are still responsible for generating knowledge of the map by keeping track of both the nodes they visited and the treasures they found. However, they now share this knowledge with other explorer agents in a distributed manner, enabling the efficient construction of a complete representation of the map across multiple machines. Once the map is fully explored, the explorer agents would initiate a random walk and interact with collector and tanker agents in a distributed way, providing them with details about the location of treasures and the routes to reach them. Collector agents are responsible for collecting and transporting the treasures, while tanker agents are responsible for covering large areas facilitating the delivery of the treasures to the collectors. This distributed architecture allows for improved performance and increased fault tolerance, as the system can continue to function even if some of the agents find some problems over the course of time. All the agents work together in a distributed manner to explore, gather, and move treasures efficiently in the map.

Some of the function and capabilities we provided to our agents can be seen in the figure

```
- ExploCoopBehaviour
  - action()
    - tmpRandomMovement(L_Obs)
      - moveToNextNodeRandom
- ShareMapBehaviour(Agent, Period,
                    Map, Receivers)
- ShareTreasuresLocBehaviour(Agent, Period,
                             Treasres, Receivers)
- SharePath(Agent, Map)
```

```
- RandomTankerBehaviour
  - onTick()
     - chooseNextNode(L_Obs)
```

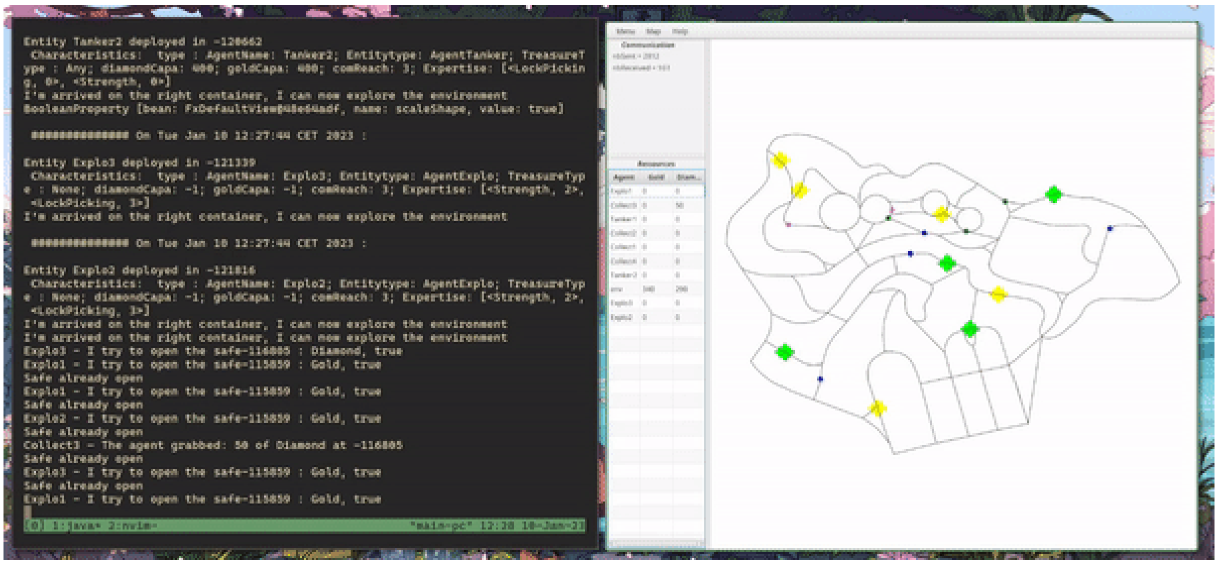
```
- CollectorBehaviour
  - onTick()
     - solveBlockedPath()
        - sendBlockingInfo()
           - getRemainingPath()
        - getBlockingInfo()
           - getRemainingPath()
     - updatePotentialTreasures()
     - receiveMission()
     - getStopMessage()
     - requestExploreHelp()
     - sendTreasureRequest()
     - backOff(L_Obs)
     - moveToNode(L_Obs)
     - moveToNextNodeRandom(L_Obs)
     - shareTreasureInfo()
     - mergeTreasureInfo()
```

Figure 12 – Agents capacities

7 Final Look of our IMAS

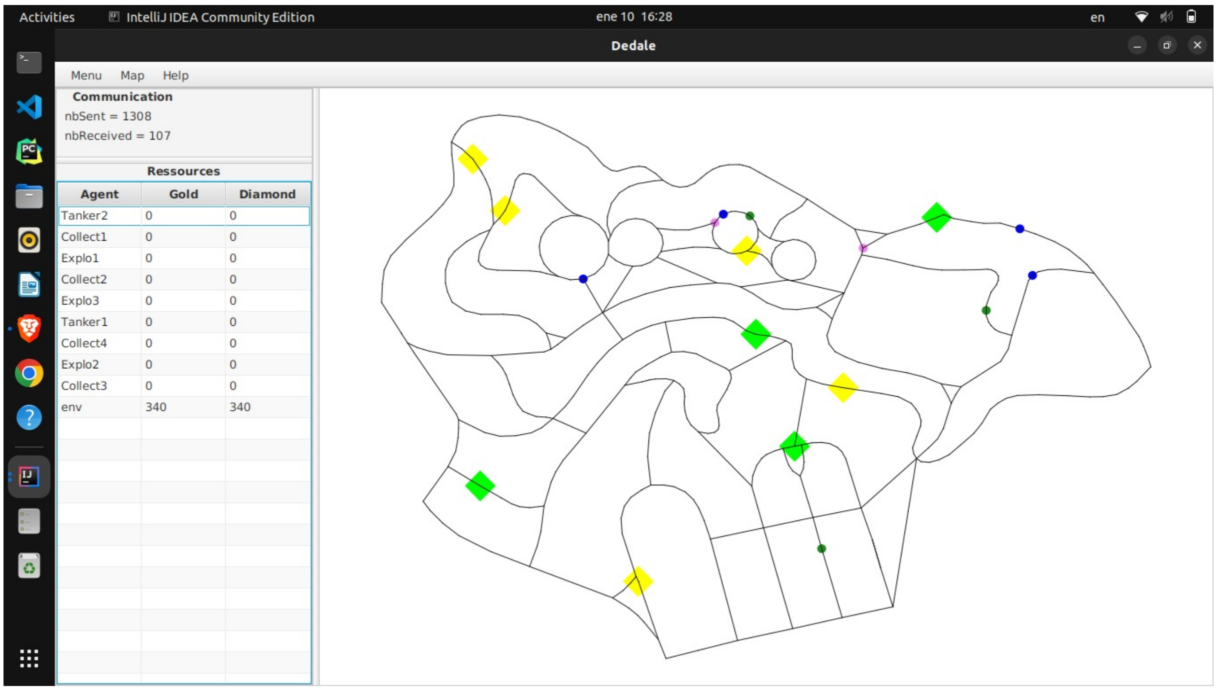
The scenario depicted in the image below represents a distributed problem-solving scenario in which multiple agents, represented by the lead explorers, collectors, and tankers, participate in a coordinated effort to locate a specific target, represented by the treasure, open it, and pick it up. The initial state of the system is characterized by a lack of complete knowledge about the environment, represented by the map of Rio de Janeiro, and explorers must actively collect information through exploration to achieve their goal as seen in the image.

The multi-agent system was implemented with different architectures such as a decentralized approach, where each agent acts independently based on its own local information.



```
Entity Tanker2 deployed in -120662
 Characteristics:  type : AgentName: Tanker2; Entitytype: AgentTanker; TreasureT
ype : Any; diamondCapa: 400; goldCapa: 400; comReach: 3; Expertise: [<LockPickin
g, 0>, <Strength, 0>]
I'm arrived on the right container, I can now explore the environment
BooleanProperty [bean: FxDefaultView@48e64adf, name: scaleShape, value: true]

 ################ On Tue Jan 10 12:27:44 CET 2023 :

Entity Explo3 deployed in -121339
 Characteristics:  type : AgentName: Explo3; Entitytype: AgentExplo; TreasureTyp
e : None; diamondCapa: -1; goldCapa: -1; comReach: 3; Expertise: [<Strength, 2>,
 <LockPicking, 3>]
I'm arrived on the right container, I can now explore the environment

 ################ On Tue Jan 10 12:27:44 CET 2023 :

Entity Explo2 deployed in -121816
 Characteristics:  type : AgentName: Explo2; Entitytype: AgentExplo; TreasureTyp
e : None; diamondCapa: -1; goldCapa: -1; comReach: 3; Expertise: [<Strength, 2>,
 <LockPicking, 3>]
I'm arrived on the right container, I can now explore the environment
I'm arrived on the right container, I can now explore the environment
Explo3 - I try to open the safe-116805 : Diamond, true
Explo1 - I try to open the safe-115859 : Gold, true
Safe already open
Explo1 - I try to open the safe-115859 : Gold, true
Safe already open
Explo2 - I try to open the safe-115859 : Gold, true
Safe already open
Collect3 - The agent grabbed: 50 of Diamond at -116805
Safe already open
Explo3 - I try to open the safe-115859 : Gold, true
Safe already open
Explo1 - I try to open the safe-115859 : Gold, true

[0] 1:java* 2:nvim-                                  "main-pc" 12:28 10-Jan-23
```

Figure 13 – Implementation of Hunt treasures



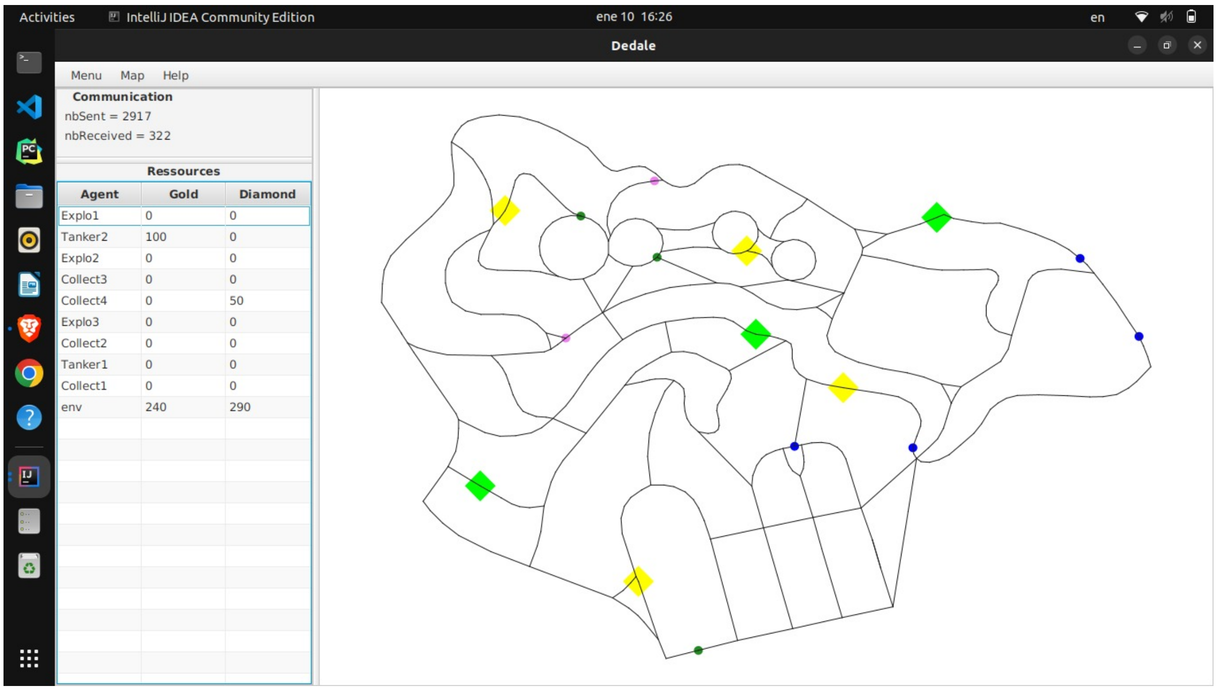
Communication
nbSent = 1308
nbReceived = 107

Ressources

Agent	Gold	Diamond
Tanker2	0	0
Collect1	0	0
Explo1	0	0
Collect2	0	0
Explo3	0	0
Tanker1	0	0
Collect4	0	0
Explo2	0	0
Collect3	0	0
env	340	340

Figure 14 – Implementation of Hunt treasures at first steps

In the following image, it is observed that the agent represented by tanker 2 is executing a task of recollecting treasures after a period of collecting and exploration, during which the explorers share information with each other and the collector bring the treasures. This behavior can be seen as an example of cooperative problem-solving among agents in a multi-agent system.



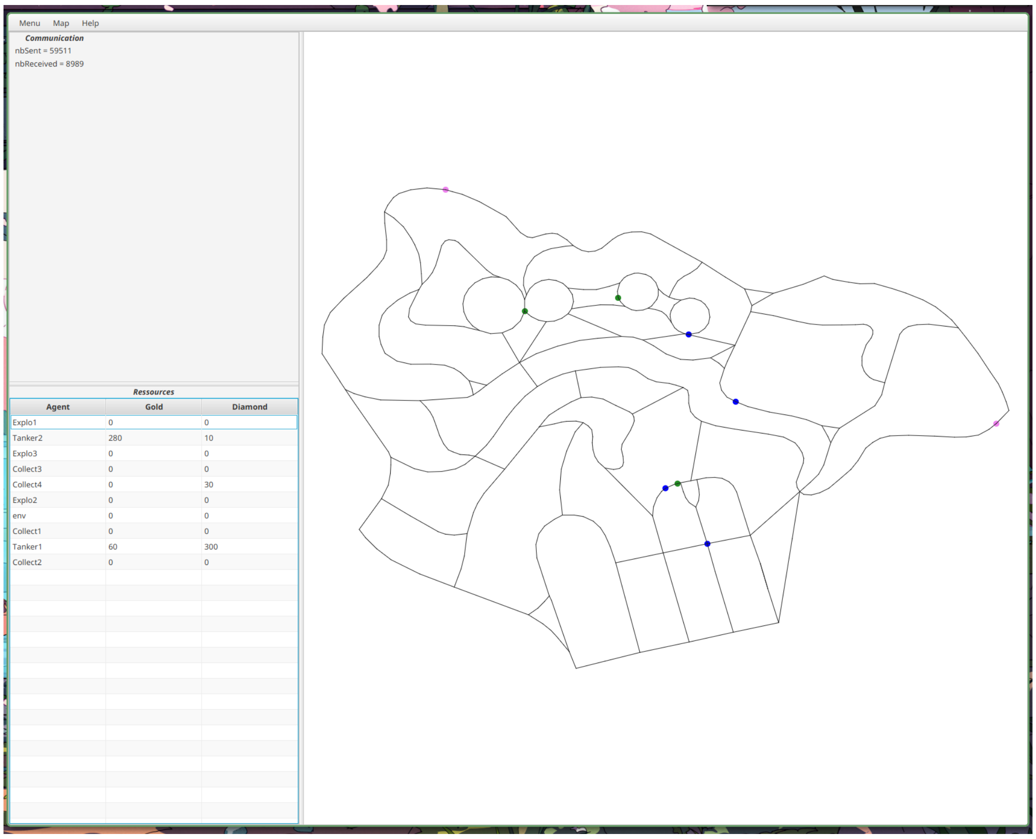
Communication
nbSent = 2917
nbReceived = 322

Ressources

Agent	Gold	Diamond
Explo1	0	0
Tanker2	100	0
Explo2	0	0
Collect3	0	0
Collect4	0	50
Explo3	0	0
Collect2	0	0
Tanker1	0	0
Collect1	0	0
env	240	290

Figure 15 – Implementation of Hunt treasures tankers recollecting

The exploration process undertaken by the agents, allows them to gather information about the location of the treasures in the environment, represented by the map. This information is then shared among the agents through a communication protocol.



Communication
nbSent = 59511
nbReceived = 8989

Ressources

Agent	Gold	Diamond
Explo1	0	0
Tanker2	280	10
Explo3	0	0
Collect3	0	0
Collect4	0	30
Explo2	0	0
env	0	0
Collect1	0	0
Tanker1	60	300
Collect2	0	0

Figure 16 – State of the environment on final steps of the simulation

From the depicted image, it can be inferred that the simulation is in its final stages. The representation shows the agents, represented by the collector and tanker, successfully completing their respective missions. This outcome is a result of the agents' ability to share information with one another, which allows them to coordinate their actions and increases the overall efficiency of the system.

The sharing of information plays a crucial role in this scenario as it enables the agents to reduce the search space for the target objects. This is achieved by reducing the uncertainty about the environment and allowing the agents to plan their actions with greater precision.

8 Testing and Challenges

The main reason of why our final design for the Multi Agent System implemented to solve the Treasure Hunt is as we present it in the previous section is mainly due the observations we got while testing such system.

At the very beginning we started with the default agents and behaviours from Dedale:

- Explorers: They were moving towards not explored nodes, following a strict path at each step. They also share information about their map representation with other explorers. Once they finish exploring, they stop and stay at the last node visited for the rest of the simulation.
- Collectors: They choose any of the nodes available randomly to move, except their current node. If they are in a treasure they try to grab it. They also try to deliver whatever they have to the tanker.
- Tankers: They choose a node randomly to move at.

So, once we added the map of Rio de Janeiro, we could start to see how these default behaviours were very limited. We will now cover the main challenges, how we overcame them and what was the performance achieved by them.

8.1 Random Movement

As said before, the random movement of the agents is based on choosing a node randomly among the observations in the current node. All nodes have the same probability to be chosen. This is a behaviour that works fine in grid type of maps, where at each node they have more than two choices to move at.

When using other maps, in this case our map Rio, the main challenge rises: straight streets are a big problem. There are some paths that can have more than 10 nodes straight without any bifurcation. This makes it very difficult for the agent to get out of such path, specially if he spawns in the middle of it. This is because, in a practical example of a path with 20 nodes, he will need to move forward or backward only 10 times to exit. This is the same as throwing a coin 10 times and getting 10 tails or heads. The probability is very low, so it is very likely that the agent never gets out of that position. This is what happened to us the first times, all agents were moving around the same area forever, not even finding in a random way any treasure.

In order to solve this we added a buffer mechanism that keeps track of the last n visited nodes and tries to avoid them. So if an agent is in a long path, he can only move forward or backward. Since the nodes behind him were already visited, he cannot go back and will always prioritize to go upfront (to not visited nodes).

Of course, this leads to potential blocking between agents when the only not visited node is being occupied by another agent. In this cases, the agent will try to move to any of the nodes available, no matter if it was visited or not, and clear its buffer so that he can take a new direction.

Thanks to this, now all the agents with random movement had a more fluid and organic movement that allowed them to cover most of the map without issues and without getting stuck.

8.2 Explorers getting stuck

The random movement is now fixed, but explorers do not behave that way during the exploration phase. They follow the shortest path from their current node towards a goal node. This can create a conflict when two explorers have crossing paths. There will be a moment that one explorer wants to go to the node where there is another explorer that wants to do the inverse.

Since explorers have map knowledge, we came up with a simple idea: reuse the random walk behaviour during a limited amount of steps and then come back to exploration. This will be only triggered when the explorer has been stuck for multiple consecutive cycles. Even though this does not guarantee they get unstuck all the time, when testing this behaviour, they never got stuck again.

This change allowed us to get from a solution where explorers never finished exploration and ended up being stuck, to a solution where explorers are not so rigid with their plans but can finish the exploration easily.

8.3 Detecting agents in close proximity

One of the problems we have encountered in knowing how to handle the interaction between agents is their ability to detect nearby agents to interact with. Given the initial constraints of the problem, the detection radius of the agents is set to zero. This implies the continuous sending of messages in search of a specific response to initialise a deeper type of interaction.

In addition to the fact that messages have to be sent continuously, the agents that should be listening for such messages must be alert all the time to catch such messages and catch a response. If such messages do not have a matching protocol or an identified number, i.e. are not properly addressed, it can cause certain messages to reach the wrong recipients, thus receiving wrong responses and causing problems in the overall course of the simulation.

To deal with this situation, we take advantage of the tools provided by Jade for sending messages. Since we configure them with specific protocols according to the type of interaction to be performed, this greatly improves the communication flow between agents and allows to coordinate interactions in a faster and easier way.

8.4 Randomness is slow

Even though if we left all the agents moving randomly in a map they will eventually collect all the treasures, this would take a huge amount of time with bigger maps. We explored how in toy, smaller maps like the grid they actually solve it very fast (some times). But for bigger maps like Rio, it could take hours until they randomly unlock, collect and deliver everything.

In order to solve this, we incorporate treasure tracking and mission requests. Collectors will keep track and share information about the position and status of the remaining treasures to collect. They don't have map knowledge, so whenever they are moving randomly and encounter an explorer, they will request them to provide the shortest path to the closest treasure he can open (same type as the collector, and with enough treasure to collect). The explorer will compute the different paths, and if he is able to provide any, he will send it back to the collector.

Now the collector is in a mission state, leaving the random movement. He has a defined path to follow and he will come back to the random behaviour once he has reached the goal and collected the treasure.

This is very helpful in advanced steps of the simulations, when explorers have all or almost all the information of the map, collectors know all the treasures left and randomness is too slow to get the final treasures left. It speeds up the simulation.

Yet, this introduces a new issue:

8.5 Collectors getting stuck on mission

Same as happened with explorers before can happen now: two paths can cross in opposite directions and they are not able to move. Now the challenge is bigger, since as the collectors don't have map knowledge, they cannot do a random walk and resume it wherever they are, because they might end up too far from the mission path to retake it.

The solution implemented here was backing off. When two collectors on a mission collide, they start sending specific messages for collectors. So they will both receive a message from the other. They both share the remaining of the mission path they have to do, and whoever has the longest path, will back off so that the other agent can proceed. If the lengths are the same, the collector with higher value of their name id will back off.

In order to back off, the agent will go back through the same nodes of the mission path until he observes a reachable node that is not in the mission path of the other agent. He moves there, waits for the other agent to move on, and retakes his mission path again.

After this, we solved all the potential conflicts in movement of the agents, so we are sure we will have a simulation that will eventually end no matter what. Since there is still some randomness involved, the time to solve the map will vary from run to run. We could not improve more from this, so future steps would be to find a way of adding better reasoning capabilities to the agents o new agents, without complicating too much the movement in the map space.

Conclusion

In this project, we have developed an Intelligent Multi Agent System for solving the treasure hunt problem, in which several agents are scattered around the map and they must cooperate between each other to gather all of the treasures in an efficient way. This project has been developed in several stages, first we started defining the environment and agent architectures, the agent properties and the overall system structure. Then, we studied several coordination and negotiation strategies for enabling the communication between the agents. Finally, we implemented the system in JADE. During the implementation, several changes were done to the structure of the proposed solution, such as simplifying the problem to fewer agent types, discarding thus coordinator agents and manager, and also modifying the communication strategies between the agents. The final result that has been obtained is satisfactory as the agents are shown to solve the treasure hunting problem by coordinating among them and showing an intelligent behaviour.

References

[1] M.Wooldridge: An introduction to MultiAgent Systems (Chapter 5)

[2] M.Wooldridge: An introduction to MultiAgent Systems (Chapter 2, section 10.3)

[3] T.Salamon: Design of Agent-Based Models

[4] M.Wooldridge: An introduction to MultiAgent Systems (Chapter 8)

[5] M.Wooldridge: An introduction to MultiAgent Systems (Chapter 14)

[6] Rieger, C., Zhu, Q. (2013, August). A hierarchical multi-agent dynamical system architecture for resilient control systems. IEEE.

[7] Unknown, (2015), ”JADE Setup for beginners”, Ideaheap, https://www.ideaheap.com/2015/05/jade-setup-for-beginners/, accessed 2nd of november, 2022

[8] Cédric Herpson. Dedale: A Realistic Simulation Framework for Decentralized Multi-Agents Problems. (to appear - draft version :hal-02392383)