UNIVERSITAT ROVIRA i VIRGILI

Introduction to Multi-Agent Systems

Final Project: Treasure Hunt Problem

Authors:

Becerra Tomé, Alberto

Campàs Gené, Carla

Bercowsky Rama, Andrés Eduardo

González Béjar, Javier

Monfort Grau, Marc

Universitat Rovira i Virgili

Master's degree in Artificial Intelligence

Tarragona, January 2024

Abstract

This project introduces a solution for a constrained map treasure hunt involving gold and diamonds, employing an intelligent multi-agent system. The task encompasses the strategic discovery and retrieval of treasures within a graph map.

The solution introduces three agent types: explorers, collectors, and tanker agents. Explorers observe the map, collecting information on treasure locations and unlocking the treasure chest for collectors to be able to access it. Collectors retrieve treasures and give them to tanker agents for secure storage.

This project has been developed using the Java programming language and utilizing the JADE (Java Agent DEvelopment Framework) library and the Dedale platform. The system design prioritizes efficiency and agility for a prompt resolution of the treasure hunt problem. This document chronicles the project phases, from initial problem analysis to the final implementation, incorporating updates aimed at refinement. It provides clarity on the problem space nuances and agent interactions.

Beyond the treasure hunt context, the solution's collaborative agent approach holds potential applications in diverse domains, such as search and rescue missions, environmental monitoring, and logistics.

Keywords: Dedale, Multi-Agent, Cooperative Exploration, Treasure Hunt, Pathfinding, Communication

Contents

1 Introduction

Multi-agent systems (MAS) are a type of artificial intelligence system that consists of multiple autonomous agents that work together to achieve a common goal. MAS are often used in complex environments where traditional single-agent systems would be inadequate.

MAS comprises fundamental entities known as agents. These computational systems strive to empower us to independently address diverse tasks. The rise of agents corresponds to the evolution of computational systems emphasizing ubiquity, interconnectedness, intelligence, delegation, and human-centric programming.

The foundational principles of MAS revolve around creating computational entities known as agents, each possessing the capabilities to operate autonomously, negotiate, and adapt to dynamic environments. These agents exhibit fundamental properties such as autonomy, reactivity, reasoning, learning, and effective communication. The project also explores diverse agent architectures, ranging from reactive systems with swift local event responses to deliberative models that engage in complex reasoning.

An agent gathers information about its environment through sensors and executes tasks autonomously through actuators. Agents must function and adjust to various types of contrasting environments, some of which they can regulate and others they cannot. These agents may have diverse architectures that dictate how the agent chooses its next move.

There are three primary types of agent architectures: reactive, deliberative, and hybrid. Reactive agents can swiftly respond to local occurrences without requiring intricate reasoning. They are straightforward and computationally efficient. However, their interactions can produce complex collective behaviors. Deliberative agents, on the other hand, have a model of the world they inhabit and can make plans to make more precise decisions. This comes at the expense of higher computational complexity. Hybrid architectures combine the strengths of these two systems.

The objective of the practical work is to develop a multi-agent system that simulates a treasure hunt using Dedale, a JADE-based agent environment framework. The treasure hunt takes place on a map represented by an undirected graph with at least 500 nodes, sourced from OpenStreetMap data and converted to DGS format. The treasures, either gold or diamonds, are dispersed across the map within safes that require certain expertise levels in strength and lock-picking to be unlocked. The agents must work together to explore the environment and find and collect the treasure.

Three types of agents are central to this scenario:

- **Explorers**, tasked with navigating and understanding the environment.
- **Tankers**, responsible for transporting the treasures.
- **Collectors**, whose role is to gather the treasures.

An optional Coordinator agent can also be introduced to facilitate communication and coordination among the primary agents. The agents must work together to achieve the common goal of finding and collecting all of the treasure. The explorers must explore the environment and find the treasure. The collectors must collect the treasure and store it in their backpacks. The tankers must find the collectors and store the maximum amount of treasure possible.

The agents can communicate with each other to share information and coordinate their actions. For example, an explorer might communicate to a tanker the location of a treasure that it has found. A collector might communicate to a tanker that it is full and needs to unload its backpack.

The challenge is to enable the agents to cooperate seamlessly, coordinate their efforts, and avoid deadlocks in their movements to optimize the treasure collection process. The design of the multiagent system will focus on creating an efficient coordination mechanism that takes into account the distinct capabilities of each agent type, ensuring effective communication and strategic movement to complete the treasure hunt in minimal steps.

The treasure hunt multi-agent system is a complex example of how MAS can be used to solve real-world problems. By working together, the agents can achieve a goal that would be impossible for any single agent to achieve on its own. In the following sections, we will discuss the different types of agents in the system, the coordination mechanisms they use, and how they work together to achieve the common goal of finding and collecting all of the treasure.

2 Proposed Multi-Agent System

2.1 Environment Properties

The environment consists of a graph of at least 500 nodes, each node representing a unique place on the map. The environment can be described by the following properties:

- **Acessible:** The agents can easily access and process all the information they need about the environment, making it accessible.
- **Deterministic:** The environment is deterministic, meaning that any action taken by the agents will have a predictable and guaranteed result. There is no uncertainty in the environment, and the agents can rely on the current state to determine the next state.
- **Discrete:** The environment is discrete, meaning that the agents can only take a limited number of actions at any given time. The agents do not have to deal with continuous variables or infinite possibilities, making the environment easier to learn and navigate.
- **Episodic:** The environment is episodic, meaning that each episode is independent of the previous one. The agent's state in one episode does not affect its state in another episode, so it can focus solely on the task at hand.
- **Static:** The environment is static, meaning that it does not change over time except when the agents take action. There are no external factors that could disrupt the agent's learning or task performance.

2.2 Map

The environment is represented as an undirected graph, depicting the geographical layout where agents can navigate. Nodes within this graph correspond to specific locations on the map, some of which may conceal valuable treasures. Meanwhile, the edges of the graph define the available pathways, enabling agents to traverse between these map nodes.

To construct this environment, we used OpenStreetMap data. Our initial map was comprised of 1101 nodes [1]. This particular map captures the area of Nou Eixample Sud in Tarragona, adjacent to the port, and encompasses a network of 296 distinct routes.

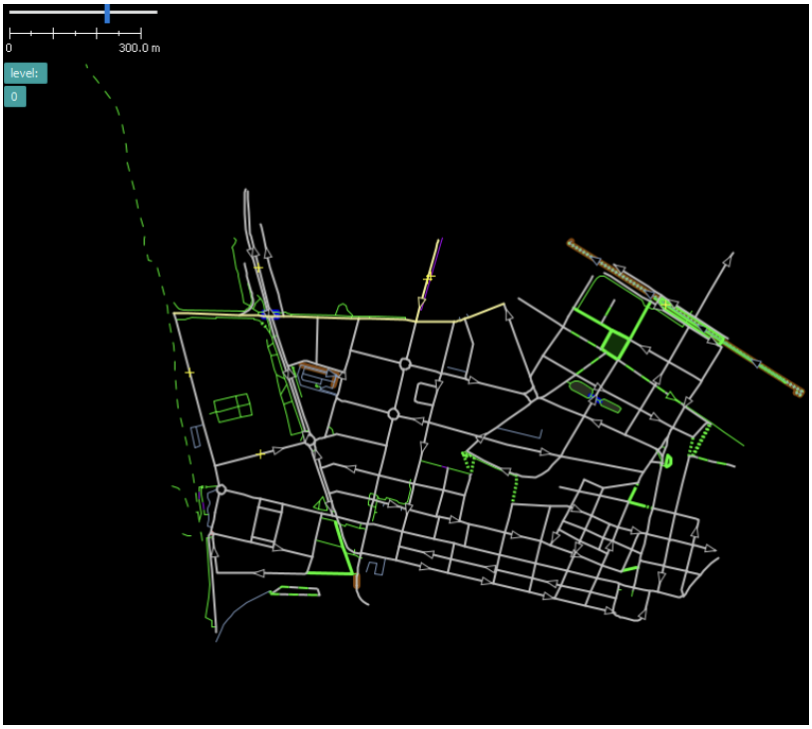

Figure 1: GeoJson Map representation

However, we found this map to be too large, therefore we decided to make this map smaller to simplify how the map was visualized and be able to see how the agents interacted in a much faster way. Furthermore, the size of our map (over double that of the initially suggested map) limited further the capability of our agents to communicate with each other. This new map is therefore constrained to the recommended size, of 500 nodes. Our final map can be seen below.

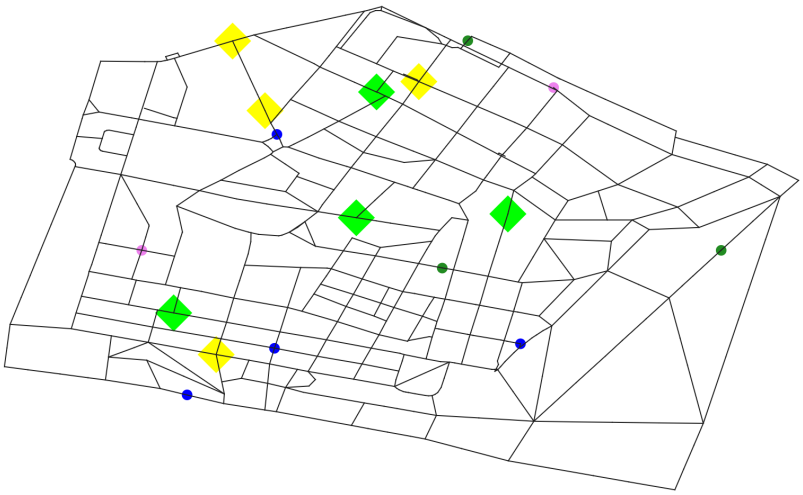

Figure 2: GeoJson Map representation

2.3 Agent Architectures

Agent selection and configuration are paramount to system success. Identifying the appropriate agent type, with clear definitions of interconnection, interoperability, and collaborative mechanisms, is essential.

Agents are commonly categorized into distinct types based on their purpose:

- **Collaborative agents:** These agents foster teamwork, negotiation, and goal alignment among agents.
- **Interface agents:** These agents excel at task automation and user-centric adaptation through learning algorithms.
- **Information agents:** These agents excel at accessing, extracting, and presenting relevant information personalized for other agents or users.
- **Facilitators:** These agents oversee the system's internal connections and orchestrate agent collaboration.
- **Translator/wrapper agents:** These agents bridge the communication gap between systems, regardless of their agent-based nature.

Custom agent types can be tailored to specific tasks.

All the base types of agents proposed by our system are collaborative agents. Different types of coordinators and other agents can be considered interface or facilitator agents.

We can also differentiate the agents by reactive, deliberative, and hybrid agents.

- **Reactive systems:** These systems excel at responding swiftly to immediate stimuli without extensive deliberation. Their simplicity and computational efficiency make them well-suited for real-time environments.
- **Deliberative systems:** These systems possess an internal representation of the world and employ planning techniques to make well-informed decisions. However, this computational overhead makes them less suitable for real-time applications.
- **Hybrid systems:** These systems combine the strengths of reactive and deliberative approaches, often with a reactive layer dominating the decision-making process. They are typically implemented as hierarchical decision layers, but a unified methodology for their development is lacking.

The collector and tanker agents are reactive agents, however, the explorer is a hybrid agent as it is reactive to the components in the map but it also deliberates the paths that the tanker and collectors have to take.

2.4 System Overview

In the domain of the treasure hunt problem, abstract agent architectures serve as a foundational framework for understanding and implementing agents' high-level components and their interactions. This framework identifies three primary agent roles: the Explorer, the Tanker, and the Collector. Each of these agents has distinct percepts, internal states, and available actions.

The Explorer is tasked with gathering information about the environment, which includes its current position and the locations of other agents. It also needs to keep track of the locations of treasures in the vicinity. The Explorer's internal state is characterized by its knowledge of the environment, which typically takes the form of a map that gets updated as the agent explores new areas. It also contains information about the actions of the collector and tanker agents, which the explorer acts as a leader for. Additionally, the Explorer maintains its current goal, which may involve discovering new territories or sharing information with other Explorers. The actions available to the Explorer are movement, enabling it to navigate to new locations; sharing its map with other Explorers to enhance collective knowledge; and computing optimal paths between nodes in the environment, which is crucial for efficient exploration as well as sharing the most optimal path for other agent types.

The Tanker plays a crucial role in the treasure hunt problem. It is equipped with percepts that include information about its current location, the positions of other agents, and most importantly, the locations of treasures. The internal state of the Tanker is primarily characterized by its cargo, representing the treasures it has collected, and its designated destination for depositing those treasures. The Tanker's actions encompass relocating to different positions within the environment, picking up treasures when it is in proximity to them, and depositing these treasures at their intended destination. The Tanker's decision-making process is shaped by factors such as proximity to treasures and the need to deliver collected treasures to the specified drop-off locations.

Similarly, the Collector operates within the treasure hunt problem with its own set of percepts, including its current location, the positions of other agents, and the locations of treasures. Like the Tanker, the Collector's internal state maintains information about its cargo, the treasures it is currently carrying, and its destination, which represents the drop-off point for the collected treasures. The Collector's actions mirror those of the Tanker, allowing it to move to new locations, pick up treasures, and deliver them to their designated destinations. Coordination with other agents is vital for both the Tanker and the Collector to ensure efficient treasure collection and delivery.

To facilitate effective coordination among these agents, a variety of mechanisms can be employed. These mechanisms include:

- **Message Passing:** Agents can communicate with one another by sending messages to share information, coordinate actions, and exchange critical data. Message passing is especially useful when agents need to provide real-time updates on their status and share important findings with their peers.

- **Leader-Follower:** In certain situations, one agent may be designated as the leader, while others follow its instructions. This hierarchical approach ensures a structured and coordinated

response, with the leader guiding the actions of the follower agents. The leader-follower mechanism is particularly useful when a clear decision-maker is needed to streamline the agents' actions.

2.4.1 Agent Types

Explorer

The explorer agent has as a goal to explore the environment and communicate their findings to other agents in the team. The explorer has to be able to quickly get around the map and update this accordingly with other explorer agents. One of their most important factors is the ability to quickly get around the map and identify the positions of the treasure (availability, quantity, etc.) within the map.

The explorer also has the best strength and lock-picking ability. This means it will be able to open most of the locked environments and make the whole map available to other agents. While at the beginning of the execution the explorer's task is to find the map and the resources in the map, eventually, the task becomes helping other agents find paths from one node to another to optimize the path taken.

Therefore, the explorer will find the resources on the map and be in charge of communicating with other agents about the most efficient way to get to a resource. These will coordinate the agents and communicate with them to be able to control how the agents move within the map. They will communicate with other explorer agents to find the most relevant information about the map. With tankers they will communicate to find a collector that needs to dump the collected treasures and with the collector they will find the best path to still existent treasure in the map. To do this, the explorer will be a lightweight system that will move fast within the map to keep agents updated with their goals.

Tanker

The tanker has as a main goal to carry the obtained treasurers (gold and diamond). They have a very large capacity to hold each treasure and therefore they are the agents that are responsible for storing the existing treasures. However, the tankers do not have the capability of collecting any treasure or opening any safes. Therefore, they require the collaboration of other agents to be able to move through the map and identify their best positions and possibilities to help collectors store their treasure.

Collector

Collectors have as a main goal to collect the treasure, however they have many limits to their capability. They are only able to collect one type of treasure (either diamonds or gold) but not both, this means that they have to take into consideration where they are headed to make sure that they can collect the correct treasure and optimize their path for this. For this, they will need the explorer They have a limited capacity of carrying the given treasure. They therefore will need to find a tanker with capacity to be able to store the obtained treasure, given that a tanker is in range

for the collector to drop off.

Coordination Agent

In the initial implementation of our agents, we do not want to include a coordination agent. This is because the coordination agents will not have the capability of assisting in the goal of obtaining the treasure and will rather get in the way of the other agents blocking possible passable nodes and making paths longer. We therefore have opted to make explorer agents distributed coordination agents. These will communicate with each other decisions and keep an updated log of communications with each other. Since we are not making one agent make all the decisions but distributing them amongst all the available explorer agents, these will be more dynamic. We will also attempt to make these decisions run in the background to maintain the explorer's goal to continuously move through the map as quickly as possible to maintain an updated version of the same.

2.4.2 Agent Properties

Explorer

The explorer is the agent in charge of controlling the actions on the map and executing these. For this purpose, an initial behavior that this agent has is to be able to scope out as much of the map as it possibly can. In other words, the explorer will have to initially visit the maximum number of nodes possible, maximizing the proportion between visited nodes and the standard deviation of the number of times a node has been visited.

$$\text{maximize} \left(\frac{\sum_{v \in V} \text{times\_explored}(v)}{1 + \text{std}(\text{times\_explored}(v) \mid v \in V)} \right)^2 \tag{1}$$

$$\text{Maximize nodes visited function} \tag{1}$$

Once a great proportion of the map has been explored and communicated between all agents, the explorer agents can then remove their initial objective to visit the maximum number of nodes possible and add a plan to make sure all agents efficiently collect and store as much of the treasure in the map as possible.

The explorer's behaviors include:

- **Say Hello**: An initial way to find all agents on the map is by broadcasting a hello message to all the agents on the map and requesting their information. This is because once we have the AIDs of each of the agents in the map we will be able to communicate with them directly given the need to. From each agent, we will need to know the type of agent they are, if they can collect any type of treasure, and their storage capacity.
- **Share Map**: The map will be shared between the different explorer agents, therefore one of the behaviors of the explorer is to give this map to other explorers. The agent will, therefore, also be prepared to receive a map from another agent and be able to update their map accordingly.

- **Share Next Steps**: The explorer agent will have to share the next steps with each collector and tanker to optimize what they are supposed to do next. This will calculate the minimum path to be taken from their initial position and allow them to find a treasure nearby.

- **Explore Behavior**: The explorer has to be able to explore the map quickly without repeating nodes. Therefore, this behavior has its function in optimizing the formula seen in Figure 1 above.

Utility Function

The utility function for the explorer can be defined based on various objectives:

1. **Maximizing Map Exploration**: The primary role of the Explorer is to uncover and explore as much of the map as possible. Thus, the utility function should encourage the Explorer to visit unexplored nodes and contribute to the team's understanding of the environment. This can be represented as:

$$\text{Utility} = \text{Weight\_Exploration} \times \text{Number\_of\_Unvisited\_Nodes}$$

2. **Effective Communication**: The Explorer needs to share information with other agents. This includes broadcasting its findings to the team, sharing maps, and coordinating with other agents. Effective communication is critical for the team's success. The utility function should promote communication, and you can define it as:

$$\text{Utility} = \text{Weight\_Communication} \times \text{Information\_Shared}$$

3. **Assisting Other Agents**: The Explorer should actively assist Tankers, Collectors, and other Explorers by providing them with information and optimizing their routes. This can be represented as:

$$\text{Utility} = \text{Weight\_Assistance} \times (\text{Number\_of\_Tankers\_Assisted} + \text{Number\_of\_Collectors\_Assisted})$$

4. **Lock Opening**: Since Explorers have better strength and lock-picking abilities, they should be motivated to open locked environments to enable access for other agents. Encouraging lock opening can be represented as:

$$\text{Utility} = \text{Weight\_Lock\_Opening} \times \text{Number\_of\_Locks\_Opened}$$

5. **Resource Identification**: The Explorer should identify the location and quantity of treasures (gold and diamonds) within the map. This information is crucial for the team's decision-making. You can use the following in the utility function:

$$\text{Utility} = \text{Weight\_Resource\_Identification} \times (\text{Number\_of\_Gold\_Spots} + \text{Number\_of\_Diamond\_Spots})$$

6. **Path Optimization**: The Explorer should work on finding efficient paths for other agents, reducing their travel time. This can be represented as:

$$\text{Utility} = \text{Weight\_Path\_Optimization} \times \text{Number\_of\_Optimized\_Paths}$$

Specific weights can be assigned to each objective based on their importance within the multi-agent system. The actual utility function would combine these objectives with their respective weights to calculate the explorer's overall utility.

Assigned Tasks

The explorer's assigned tasks include:

- Explore and maintain a real-time state of the map.
- Communicate with other explorers to exchange new findings on the map.
- Communicate with tankers to determine their positions and guide them to collectors.
- Communicate with collectors to guide them to collectible treasures (diamonds or gold).
- Open locks to provide other agents with easier access to the map.

Tanker

The tanker is in charge of collecting as many treasures from the collectors as possible. This way, the collectors will be free to take as much treasure from their next target as possible. To facilitate this, the tanker should have as a main objective to stay close to treasure nodes that collectors will go to, and take into account, wherever possible, which collectors have a full backpack.

Once they are adjacent to a collector they can communicate with the collector to what their maximum offload capacity is to make sure that the collector gives them an adjusted value.

The tanker's defined behaviors include:

- **Say Hello**: An initial way to find all agents on the map is by broadcasting a hello message to all the agents on the map and requesting their information. This is because once we have the AIDs of each of the agents in the map we will be able to communicate with them directly given the need to. From each agent, we will need to know the type of agent they are, if they can collect any type of treasure, and their storage capacity.
- **Collection Communication**: Once the tanker is adjacent to a collector it will be able to communicate the amount of treasure the tanker can intake. With this information, the collector will be able to offload the desired treasure in the tanker.
- **Tanker Behavior**: The tanker will have to find the collector that they have the capacity for to be able to take the treasure for them for the collectors to be able to take the rest of the

treasure from the map.

Utility Function

The tanker's utility function can be summarized in the following objectives:

1. **Efficiently Store Collected Treasure**: Maximizing the capacity to store and transport treasures collected from collectors:

$$\text{Utility} = \text{Weight\_Treasure\_Storage} \times \text{Amount\_of\_Treasure\_Stored}$$

2. **Facilitate Collector Movement**: Minimizing the need for excessive movement on the part of collectors by staying close to them:

$$\text{Utility} = \text{Weight\_Collector\_Proximity} \times \text{Number\_of\_Collectors\_Nearby}$$

3. **Communicate Maximum Offload Capacity**: Communicating the tanker's maximum offload capacity to collectors efficiently:

$$\text{Utility} = \text{Weight\_Offload\_Communication} \times \text{Number\_of\_Successful\_Communications}$$

4. **Minimizing Steps:** Encouraging tankers to choose paths that require the least amount of steps:

$$\text{Utility} = \text{Weight\_Minimizing\_Steps} \times \left(\frac{1}{\text{Number\_of\_Steps\_Taken} + 1} \right)$$

Assigned Tasks

The tanker's assigned tasks include:

- Storing treasures collected by collectors.
- Finding collectors to offload their treasures and avoiding unnecessary collector movement.
- Communicating with collectors to determine and optimize offload capacity.
- Communicating with the explorer to assist in their exploration efforts.

Collector The collector is in charge of getting to the safes with the treasure. These safes will either be opened by the explorer or by the collector depending on whether the collector has enough strength to open the lock. The collector will therefore have to find the treasure that it can pick up and collect it. When they are full, they will also have to find the tankers to offload the treasure if they have not found them previously.

The collector's defined behaviors include:

- **Say Hello**: An initial way to find all agents on the map is by broadcasting a hello message to all the agents on the map and requesting their information. This is because once we have the AIDs of each of the agents in the map we will be able to communicate with them directly given the need to. From each agent, we will need to know the type of agent they are, if they can collect any type of treasure, and their storage capacity.

- **Collect Behavior**: The collector will have behaviour to be able to follow the path (communicated by the explorer) to find a treasure nearby. Once found they will collect the treasure given their maximum capacity.

- **Offload Behavior**: Once the collector receives a signal from the adjacent tanker they will be able to offload the amount of treasure communicated by the tanker before they keep finding the treasure.

Utility Function

The collector's utility function can be summarized with the following objectives:

1. **Efficient Treasure Collection:** Maximizing the amount of treasure collected efficiently:

$$\text{Utility} = \text{Weight\_Treasure\_Collection} \times \text{Amount\_of\_Treasure\_Collected}$$

2. **Lock Picking (Given Strength Restrictions):** Motivating collectors to open locked safes when their strength allows:

$$\text{Utility} = \text{Weight\_Lock\_Picking} \times \text{Number\_of\_Locks\_Picked}$$

3. **Offload Treasure with Tankers:** Efficiently offloading treasures to tankers when their backpacks are full:

$$\text{Utility} = \text{Weight\_Offload\_With\_Tankers} \times \text{Number\_of\_Successful\_Offloads}$$

4. **Minimizing Steps:** Encouraging collectors to choose paths that require the least amount of steps:

$$\text{Utility} = \text{Weight\_Minimizing\_Steps} \times \left(\frac{1}{\text{Number\_of\_Steps\_Taken} + 1} \right)$$

5. **Staying Near Tanker when Backpack is Full:** Ensuring collectors stay in proximity to tankers when they cannot collect more treasures:

$$\text{Utility} = \text{Weight\_Staying\_Near\_Tanker} \times \frac{1}{\text{Distance\_to\_Nearest\_Tanker} + 1}$$

Assigned Tasks

The collector's assigned tasks include:

- Collecting treasure.
- Picking locks when their strength allows.
- Offloading treasure with tankers when their backpacks are full.
- Communicating with tankers to coordinate treasure offloading.

2.5 Deadlock Problem Solution

In the context of the treasure hunt simulation involving multiple agents, deadlocks can occur when two or more agents end up in a state where they are waiting for each other to take action, leading to a situation where no progress is made. This could happen, for example, when multiple agents are trying to access a single node or when they are waiting for resources that are held by each other. Below is a proposed solution to prevent and resolve deadlocks in the system.

To effectively avoid deadlocks in a multi-agent system designed for a treasure hunt problem, it's imperative to establish a clear hierarchy of agent priorities based on their roles. In this context, explorers should be accorded the highest priority, followed by collectors, and then tankers. The order established is due to the requirements of agility that each of the agents has. The explorers have the highest priority because they require fast movement around the map to maintain an updated image this. The next priority is given to collectors, these have an important task in finding the treasure within the map. If the tanker and the collector are in a deadlock, the tanker will initiate a conversation with the collector to offload the given treasure, and after receiving the treasure the tanker should step back. Finally, the tanker will have the last priority position.

In the case that two of the same type of agents find each other in a deadlock, we will find the lowest AID between the two agents and this one will have passing preference.

Agents must also maintain awareness of their status, including tasks, positions, and any agents they are waiting for. A well-defined communication protocol should be in place to facilitate the exchange of this critical information, allowing agents to broadcast intentions and request assistance as needed. In this case, the agents should also communicate their intention to retreat. If the agent notices a deadlock but does not receive a communication, it will know that there is no intention to retreat from the opposite. In that case, the agent will also be able to retreat to not have a permanent deadlock. This will serve as a fail-safe in case the implementation fails at any point.

3 Communication, Coordination, and Negotiation

The activity has many implications concerning the communication of the agents involved in the environment. This includes communication between the different types of agents, for example, to enable the explorer to communicate the best possible path to take to each agent.

This activity is a Cooperative Distributed Problem Solving (CDPS) project, this means our agents will be built as benevolent agents rather than self-interested agents. These agents will be built to help each other achieve a common goal: e.g. the best interest of the agents is our best interest.

If our agents were to be placed in a competitive platform, where other teams had different agents within this platform and whoever got more treasure won the game, we would have to create a self-interested agent. This would allow us to apply the competitive angle to our agents.

To select the best possibilities for communication, coordination, and negotiation for agents we will initially explore the different types of communication categories, once these have been explored we will reason the need for each of these in our platform and select the communication that best fits our goal.

Through the following sections, we will address the steps involved with task and result sharing.

3.1 Levels of Agent Coordination

We will split the level of agent coordination between low-level coordination, mid-level coordination, and high-level coordination. Low-level coordination involves the coordination of individual actions. For example, two agents might need to coordinate their movements to avoid colliding with each other.

3.1.1 Low-level coordination:

- **Avoiding collisions:** The agents need to avoid colliding with each other while moving around the environment. For this, we implement the deadlock solution if two agents are headed in the same direction.
- **Sharing information:** The agents need to share information, such as their locations, the locations of treasures, and the state of safes. The mapped version and positions of each agent will be known by other agents by sharing information. A clear example of this is sharing the map between agents.
- **Negotiating access to resources:** The agents need to negotiate access to resources, such as safes and treasures. This will prevent too many agents from going for one specific treasure on the map and giving access to taking the treasure.

3.1.2 Mid-level coordination:

- **Coordinating exploration:** The agents need to coordinate their exploration of the environment to avoid wasting time and effort.
- **Coordinating planning:** The agents need to coordinate their plans to collect treasures as efficiently as possible.
- **Coordinating task allocation:** The agents need to coordinate the allocation of tasks, such as exploring, collecting, and transporting treasures.

The coordination tasks that we see in this level will initially be run by the explorer, however, we will ponder the requirement of adding a new agent type as a coordination level between all of the other agents.

3.1.3 High-level coordination:

- **Developing a shared strategy:** The agents need to develop a shared strategy for collecting all of the treasures in the minimum number of steps possible.
- **Adapting to changes in the environment:** The agents need to be able to adapt their strategy to changes in the environment, such as the discovery of new treasures or the appearance of obstacles.
- **Resolving conflicts:** The agents need to be able to resolve conflicts that may arise, such as when two agents want to collect the same treasure at the same time.

The specific coordination mechanisms that are used at each level will depend on the specific requirements of the treasure hunt problem. For example, low-level coordination can be achieved using simple rules, such as "always yield to traffic coming from the right." Mid-level coordination can be achieved using more complex mechanisms, such as negotiation or auctioning. High-level coordination can be achieved using even more complex mechanisms, such as game theory or machine learning. Once we have a base for the agents we will be able to implement these mechanisms in a finer way and control which of these are better suited for our problem type.

3.2 Problem Decomposition

The problem decomposition involves dividing the larger problem at hand into smaller tasks amongst the distributed agents.

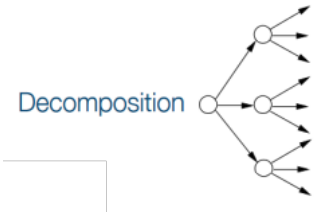

Figure 3: Problem Decomposition

Problem decomposition is a hierarchical process where the problem is divided into subproblems and the subproblems are again divided until we get a granularity that is ideal for our agents to be able to handle as single instructions in the system.

Following our initial proposal for implementation, we will decompose the problem using the explorer agents. Each of these will be able to communicate their solution while communicating with the other explorer systems.

3.3 Sub-problem Solution

The sub-problems are then to be solved to maximize our agents' capabilities. We have to keep in mind a few different aspects to make sure that the solution we obtain makes sense for all the agents and optimizes the work to be done. We do this by maximizing the coherence metric in our system.

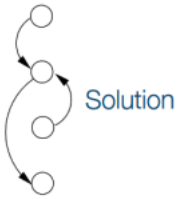

Figure 4: Problem Decomposition

Following our initial proposal for implementation, the explorer agent will trace the best possible paths with other agents. Each of these will be able to communicate their solution while communicating with the other explorer systems.

These solutions have to be consistent and make sure that each agent to whom we give instructions has a common goal. If these agents overlap in what treasure they are obtaining it will make our system inefficient and complicate it. Therefore, there has to be constant communication between our agents to make sure that the plan of each of these agents makes sense.

3.4 Answer Synthesis

There are two activities to keep in mind when speaking about solution synthesis:

- **Task Sharing:** The sub-problems should be shared with the other agents in the form of tasks. In our system, the explorer will share the task with the collector and tanker agents.

- **Result Sharing:** Once the collectors have collected their items they have to share the status back to the explorers or nearing agents to make sure that these know the most current status of the treasures.

These two communication techniques will need to be achieved to make sure that the problem solution has been achieved. This is the inverse of decomposing the problem: We solve the problem one sub-problem at a time and join these recursively to solve the larger problem.

3.5 Coordination Techniques

Coordination in multi-agent systems refers to the ability of multiple agents to work together in a way that achieves overall system goals or objectives. In a multi-agent system, individual agents are autonomous entities with their own goals, capabilities, and possibly different perspectives or interests. Effective coordination ensures that these agents can collaborate, share information, and make decisions in a manner that optimizes the overall system performance.

First, we will be explaining the theory of different coordination techniques studied. Then we will introduce the ones we plan to implement in our system, to help us solve the treasure hunt problem efficiently.

3.5.1 Voting

In a horizontal structure devoid of central coordinators, the voting method undergoes adaptation to encourage collaborative decision-making among agents. This protocol entails agents collectively determining the optimal action from a set of options, with the most voted action subsequently selected.

Within this framework, voting primarily serves to establish the order in which treasures are to be collected. Agents, informed by explorers about available treasures, participate in a voting process among collector agents to ascertain the initial treasure for collection.

The voting process in this context lacks a central coordinator, and instead, collector agents collectively initiate and engage in the voting process. Each agent independently decides and votes based on considerations such as proximity to treasures and current availability, with a preference for dense and unoccupied areas.

Before voting, all agents are informed about the ongoing process and the available options. Each agent then independently decides and communicates its choice to the group of agents, fostering a decentralized approach to decision-making.

A Plurality protocol is employed in this voting process, allowing each agent to assign one vote to its preferred alternative. The alternative garnering the highest number of votes is selected as the decision. It is assumed that agents truthfully vote based on their actual positions within the system.

In instances where a tie occurs between two or more options, agents can employ strategies such as random selection or seek a second opinion from other agents.

The final decision and execution of the chosen treasure collection involve agents collectively consolidating the results, determining the final decision, and proceeding with subsequent actions. The selected treasure is then assigned to a collector agent for collection.

This adapted voting method adheres to a fully decentralized approach, allowing agents to autonomously participate in the voting process without reliance on central authorities. It facilitates effective decision-making in the dynamic context of treasure collection within a multi-agent system with a horizontal structure.

3.5.2 Auction

Definition (McAfee and McMillan, JEL 1987): A market institution with an explicit set of rules determining resource allocation and prices based on bids from the market participants.

Agents that receive the announcement decide for themselves whether they wish to bid for the task. Factors:

- agent must decide whether it is capable of expediting the task;
- agent must determine quality constraints and price information (if relevant).

In a horizontal hierarchy where there is no central coordinator, the process of coordinating actions among agents can be distributed across the network of agents themselves. Each agent has a level of autonomy and can communicate directly with other relevant agents without relying on a central coordinator. Here's how the scenario might be adapted in a horizontal hierarchy:

When an explorer discovers a treasure and needs a collector to retrieve it, the explorer broadcasts a message to all nearby agents, including potential collectors. This message contains information about the discovered treasure, its location, and the type of treasure. Agents in the vicinity, including potential collectors, receive this message.

If the explorer can open the chest independently, it proceeds with its journey. However, if the assistance of other agents is required, the explorer waits at the treasure's location for a collector to come into the appropriate range. The collector, upon receiving the message, assesses its ability to assist and, if willing, collaborates with the explorer to open the treasure.

To coordinate the collection process, the explorer communicates its need for a collector directly to potential collectors in the area. Collectors can then submit their bids or express their willingness to assist. The explorer evaluates these bids and collaborates with the selected collector.

Similarly, the tanker's role in transporting the gathered treasure can be decentralized. The explorer communicates its need for a tanker to transport the treasure directly to nearby tankers. Tankers submit their bids or express their willingness to transport the treasure, and the explorer selects the appropriate tanker.

In a horizontal hierarchy, communication and coordination occur directly between agents, fostering a more decentralized and collaborative approach without the need for a central coordinator.

In our horizontal structure, the rules governing participation in an auction can be adapted to the decentralized nature of interactions among agents. Here's how the rules might be modified:

1. **Admission Rule:** In a horizontal structure, the admission rule can be adapted to allow agents of the same type as the initiator (who calls for the auction) to participate. Agents within proximity or with relevant capabilities can autonomously decide to participate in an auction called by another agent.

2. **Interaction Rule:** Agents, instead of sending their proposals directly to a central coordinator, can communicate directly with the agent who initiated the auction. This fosters a peer-to-peer interaction model, where agents negotiate and submit their proposals directly to each other.

3. **Validation Rule:** The validation rule can still be maintained, but agents can independently assess their availability and lack of major occupation at the time of bidding. They don't need a central coordinator to validate their eligibility; rather, agents autonomously ensure they meet the criteria before participating.

4. **Outcome Decision:** In a horizontal structure, the decision-making process for the outcome of the auction is distributed. The initiator of the auction evaluates the proposals received directly from participating agents and makes a decision based on the best bid. There is no central coordinator; instead, the decision-making authority is distributed across the network.

5. **System Flexibility:** Including auctions in our system still provides flexibility, but in a more decentralized manner. Each auction is initiated by an agent and involves direct interactions between interested agents. This approach reduces the time needed to make decisions in specific scenarios, as agents can autonomously engage in auctions without waiting for a central coordinator.

6. **Auction Scope:** In our horizontal structure, each auction involves only one instance at a time, aligning with the decentralized nature of our system. Agents autonomously engage in one-on-one auctions, enhancing the simplicity of assigning activities to specific agents based on their attributes and capabilities.

3.5.3 Organisational Structures

Organizational structures are a way to coordinate and communicate among agents in a multi-agent system. They can be thought of as a set of rules and norms that govern the behavior of the agents. These rules and norms may be explicit or implicit, and they may be enforced by the agents themselves or by an external entity.

Organizational structures can provide several benefits for multi-agent systems. They can help to ensure that the agents are working towards a common goal, that they are communicating effectively,

and that they are resolving conflicts fairly and efficiently.

There are many different types of organizational structures, but they can be broadly categorized into two groups: centralized and decentralized.

- **Centralised organizational structures**: have a single agent that is in charge of making all of the decisions. This can be effective for simple tasks, but it can be difficult to scale to large systems.

- **Decentralised organisational structures**: have multiple agents that make decisions independently. This can be more difficult to manage, but it can be more flexible.

3.6 Agent Coordination

For this system, we will have to consider which coordination and communication techniques to use throughout its implementation. For this, we will go through each situation and task assigned to each agent and figure out the best communication technique.

In the previous documentation, we expressed the agents we would be using to organize our system. In this case, we will only use the base agents explorer, tanker, and collector to complete the task at hand. We won't be adding any extra agents to compute any more logic. As we had previously defined, we will have an organizational hierarchy with a decentralized organizational structure. Each of the explorer agents will have the power to make decisions about where each of the coordination agents should be going. The other agents will have to ask the explorer agents for a task when they require one.

The initial task will be discovering the agents within the map, for this purpose we will assign an initial one-shot behavior that sends a message to all the agents in the broadcast to announce the agent's AID and information (agent type and capacities) to the other agents in range. For this purpose, we will require a larger range than the rest of the messages involved in this study. To minimize this we can implement two initial behaviors: The first one sends the agents AID and information. Once the agent has gone through a determined number of milliseconds without receiving an AID (this will depend on tests performed once the agents are up and running) the second behavior is triggered. Since we know the number of agents in our system, we can implement this as a ground truth and expect our AID list to have 9 agents (including ourselves). If this number isn't reached we can ask those agents in range for their agent list to detect all agents in our platform.

The next step is assigning tasks to all the agents. The explorer, collector, and tanker agents will all start with an initial explorer behavior. These will be tasked to walk around the map and perform corresponding actions.

3.6.1 Explorer

The explorer will quickly move around the map trying to find the treasures around the map and get a comprehensive idea of all the nodes within the map. To discover the map efficiently we will have an initial exploration period to make sure that the subsequent treasures it finds can be properly

fulfilled by tankers and collectors. In this initial behavior, it will quickly discover the map and try to open the treasures it goes finding. If the explorer finds a treasure and can open it, it will be assigned the task of auctioning off this treasure. However, auctions will not happen until the explorer's behavior has finished.

During the explorer behavior, the explorer will also periodically share the map instance with the rest of the explorers in range. The period in which the explorers will share their maps will be investigated once an initial system is up and running. This will make it simpler to fulfill the map instance with all the nodes. The explorer will therefore take the map and get the most updated version of the map by checking the timestamp assigned to each node. It will also join nodes that it had not previously discovered.

The condition for the explorer behavior to finish will be investigated once an initial system is up and running. When the explorer's behavior has finished, the explorers are assigned to auction off the treasures it has been able to open. There will be two auctions, one for the collector and one for the tanker. Initially, the explorer will be tasked with finding a collector for a specific treasure (given the treasure type). The explorer will start an auction and receive the position of the collectors that are within range. It will then assign the task of collecting the treasure to the collector that is closest to the given treasure. In the case there are no available collectors in range, the explorer will move and restart the auction elsewhere. Once the explorer has been able to locate a collector for a given treasure, it will auction off the treasure to the tankers, again based on minimum distance. In this case, we could have considered many of the protocols explained in the previous sections, however, they would have been difficult to adapt to our system due to the constraint that only the explorer has the full map and visibility of the area.

Once an auction is assigned to either a collector or a tanker the explorer will return the path towards the position the agent has to take. In the case of the tanker, it will also share the AID of the collector who will be collecting the treasure in that position. The fact that the explorer can walk about and restart the auction means that the range of interaction can be less in these stages as it will try again in the future.

Once the explorer has assigned each of the treasures it has opened to a *collector, tanker* pair, it has then fulfilled the task.

3.6.2 Collector

The collector will be initially tasked with an explorer behavior. In this initial stage, when the collector finds a treasure it can open by itself (or can take it fully), it can take this treasure without the need to wait for an explorer. In this case, it will broadcast a message to explorers telling them it has taken this treasure. If the treasure cannot be taken or opened, it will go about with the explorer behavior.

The collector will wait to receive an auction from the explorers. Once these auctions are received, they will move to the assigned slot take the treasure that is stored there, and offload to the assigned tanker. The collector will have to communicate with the tanker once this one arrives to know who and where to offload the treasure. Once the treasure is offloaded it will be freed up again to take

more treasure and repeat the action. It will then announce to the tanker that all the treasure has been taken.

Once the collector receives no more auctions, it has fulfilled its task.

3.6.3 Tanker

The tanker will be initially tasked with an explorer's behavior. This means it will move around the map randomly. It will also be subscribed to auction messages from explorers. Once a tanker wins an auction, it will be tasked with arriving at the location given a path. Once the tanker arrives at a specific location, it will announce itself to the collector assigned to take the treasure and wait for the collector to announce that all the treasure has been taken.

Once the tanker receives no more auctions, it has fulfilled its task.

4 System Implementation

In the initial phase, we systematically designed various elements of our multi-agent system specifically tailored for the treasure hunt challenge. These elements included the agent architecture, agent attributes, communication protocols (encompassing coordination and negotiation strategies), and the expected behaviors associated with each agent type. Conceptually, our design was methodically organized and free from ambiguities. Nevertheless, during the implementation phase, it became evident that the architecture we had devised manifested an unnecessary degree of complexity, ultimately compromising its efficiency in achieving the task at hand. This was mainly based on the coordination bottleneck that we had previously proposed. Due to this, we had to change the way that the system functioned. However, once our initial system was fully implemented and tested we have a short amount of time to change this. Therefore, our final system is somewhat more simple than we had initially hoped for. Nevertheless, this system proposes a very efficient form of communication and interaction between the agents.

4.1 Reevaluating the Architectural Design

Since we hadn't initially implemented our system with any other agents (e.g. communication agents, etc.) we implemented the explorer as a one-way deliberative and reactive agent. We had assigned a large number of tasks to the explorers, this meant that the explorers after their initial exploration, when they had to auction off the position of collector and tanker of one of the specific treasures we saw that the computation time taken for this was far too long, between computing all the shortest distance paths, auctioning them and receiving a final response. Our tests showed that most of the time for our agents was lost in this auctioning step. These are the main issues that we identified:

- Explorers lost a lot of time on repeated computations of paths and relaunching auctions.
- Many messages were lost between tankers, collectors, and explorers due to the block that was imposed during auctions.
- The chosen actors were very far away from the treasure since the auction was not started immediately when finding the treasure, the collectors and tankers found were usually not the most optimal. This also made it harder for the explorers to contain the proper path.

Auctions and bids are frequently employed in Intelligent Multi-Agent Systems for resource allocation or task distribution among agents. Nevertheless, the adoption of these mechanisms necessitates extensive and ongoing communication among the agents. This poses a challenge in our system, given the agents' communication range limitations and their constant mobility. Enforcing such behaviors might prove unnecessary for addressing the specific challenges posed by the treasure hunt problem we are aiming to solve. Moreover, alternative communication or negotiation techniques, such as social laws and agreements, could be more apt, depending on the nature of the problem, to facilitate effective coordination among agents.

The revised architecture adopts a streamlined structure, enhancing communication efficiency and expediting decision-making processes. Notably, the utilization of Dedale's tools ensures that the effectiveness of agents' decisions in completing subtasks remains unaffected. Within this updated

framework, the collectors, supported by explorers, stand out as agents possessing superior coordination and problem-solving capabilities. Tankers, on the other hand, focus on covering extensive map areas, strategically positioning themselves to be readily located by collectors responsible for treasure delivery.

This structure means that the collectors will communicate directly with the tankers and the explorers. Where the collectors will be able to look for tankers and communicate their intention to deposit treasure. Collectors will also communicate (bi-directionally) with the explorers to be able to find paths to treasures efficiently when required.

4.2 Agent Modifications

4.2.1 Explorers

4.2.2 Map Communication

The initial role of explorer agents is to generate knowledge about the map for sharing with other agents, while also maintaining a representation of the explored world. This will is also required for path sharing with the collector and tanker agents to provide proper communication between the agents and make the subsequent agents more efficient. Consequently, explorer agents, during their exploration, record both the nodes they visit to construct a graph representing the map and the locations of treasures they discover along the way.

In the course of map exploration, explorer agents may encounter fellow explorers within their communication range. Upon such encounters, they possess the capability to exchange knowledge, enabling the sharing of the constructed maps. This feature eliminates the need for every explorer to traverse every map node independently for a comprehensive map representation. Furthermore, the sharing of maps between explorer agents results in the expedited completion of the exploration phase.

4.2.3 Explorer - Collector Communication

As explained previously, the initially proposed communication made both the explorer and the collector collapse by the amount of incoming and outgoing messages received. Therefore, the initial implementation has to be changed to a more simple logic that allows our agents to be in constant movement.

After completing the exploration of the map, the explorer agents initiate a stochastic traversal across the nodes of the map. During this traversal, interactions occur between the explorer and collector agents in the following manner: The collector agents transmit a request to the explorer, seeking information on the location of a specific type of collectible treasure. Subsequently, the explorer agent accesses its compiled data regarding treasures and nodes, offering the collector agent the shortest path to the relevant collectible. This process repeats as the random walk recommences, with further interactions taking place whenever the explorer encounters another collector along its trajectory, facilitating the exchange of information about additional treasures.

4.2.4 Final Implementation

The algorithm for Explorer Agent behaviors unfolds in a continuous mission loop. The agent diligently traverses the map, using the formula proposed above, to ensure that we are accessing the node that has been least visited. Upon completing the map exploration, it transitions into a random walk mode. During this random walk, encounters with other agents are managed. If the Explorer meets a Collector, it responds by providing the Collector with the shortest path to a requested treasure. Similarly, if the Explorer encounters a Tanker during its random walk, it imparts information about a location near a treasure. To address situations where the Explorer becomes stuck, the algorithm incorporates a mechanism for the agent to undertake a 10-step random walk and subsequently resume the search for unexplored nodes. This iterative process continues as long as the mission persists.

4.3 Collector

4.3.1 Expected Behavior

During each clock tick, the Collector goes through a couple of routine tasks. Firstly, it checks the current node for a suitable treasure it can collect. If it finds one and has the right skills, it unlocks and stashes it in its backpack.

Additionally, as the Collector moves through nodes, it keeps an eye out for Tanker agents within its communication range. When it spots one, the Collector tries to hand over any gathered treasures, streamlining the resource transfer process as it goes about its tasks.

4.3.2 Random Walk

Initially, Collector agents move randomly, but this approach presents challenges, causing the agent to oscillate between forward and backward movements without substantial progress. To remedy this, a movement optimization function has been implemented. This function tracks the agent's recent nodes, avoiding revisits unless necessary due to obstacles or the need to backtrack. This refinement ensures a more purposeful and efficient movement along the designated path.

4.3.3 Conflicting Paths

Our hierarchy idea served its purpose between the different types of agents. However, we had to revisit the idea of the same type of agents. For this purpose, we have implemented a new technique that takes into consideration how close a collector is to their destination.

To determine priority, our implementation involves the sharing of paths between two agents. The agent with fewer remaining nodes in its path is granted higher priority. Consequently, the agent with lower priority takes the necessary steps backward, allowing the higher-priority agent to proceed along its path until completion or until encountering another agent. Subsequently, the priority measurement process recommences.

4.3.4 Collector - Explorer Communication

The explorers systematically traverse the map, documenting the treasures they discover in a list. This record includes details such as the type of treasure and the corresponding node where it is situated. When a collector encounters an explorer, the collector solicits the shortest route to a collectible treasure. Subsequently, the explorer provides the collector with a designated path, which the collector then follows. The subsequent task involves locating a tanker, stationed at a predetermined node and awaiting the collectors. It is important to note that if there is no fixed tanker at that node, the collector must initiate a random move to search for one.

4.3.5 Collector - Tanker Communication

The collector agents meticulously document their findings, maintaining a detailed inventory that includes both the type and quantity of each treasure they come across. As these collectors traverse the map, encounters with fellow agents prompt the exchange of messages containing the valuable information recorded in their respective treasure lists. Through this mutual sharing, agents update their records, incorporating new details about treasures scattered throughout the map that may have been overlooked during their individual explorations.

Despite the collaborative nature of this information exchange, there exists the possibility that, by the time both agents update their notebooks, another agent might have already traversed those locations, collecting the treasures entirely or partially. This inherent challenge underscores the significance of sharing treasure information between agents. In instances where discrepancies emerge, such as one agent noting a specific quantity for a particular treasure while the other's shared information suggests a different quantity, a collaborative correction mechanism is activated. The agent with more accurate information amends their notebook, ensuring that the collective knowledge remains precise and reflective of the dynamic state of the map and its treasures. This cooperative approach enhances the overall effectiveness of the agents in navigating and extracting resources from the map.

4.3.6 Final Implementation

The Collector Agent's behavioral algorithm is structured around a continuous mission loop. Within this loop, the agent engages in a random walk along the map, exploring various nodes. Upon encountering an Explorer during its traversal, the Collector initiates communication, requesting the shortest path to a treasure. Following the Explorer's guidance, the Collector then moves to the designated node and collects the treasure. Similarly, when the Collector meets a Tanker, it unpacks the accumulated treasures onto the tanker.

In the event of a collision with another agent, a set of actions unfolds. The Collector and the other agent exchange paths and compute priorities based on the number of nodes remaining in their paths. The agent with fewer nodes left gains higher priority. If the Collector does not hold priority, signifying a greater number of remaining nodes in its path, it takes corrective measures by adjusting its movement to distance itself from the other agent, thus avoiding a collision. This iterative process continues as long as the mission persists, guiding the Collector through dynamic interactions and collision avoidance to achieve its objectives effectively.

4.4 Tankers

4.4.1 Expected Behaviour

The primary goal of the Tanker agents is to retrieve treasures discovered by other collector agents, particularly when the collectors lack the necessary storage capacity in their backpacks. At the commencement of the simulation, all tanker agents exhibit a randomized movement pattern. This random movement persists until the explorer agents complete the exploration of the map, culminating in the generation of comprehensive knowledge encompassing all nodes and edges within the map.

4.4.2 Explorer Communication

Upon the completion of the exploration phase the explorers, initiate a random walk. At a certain juncture in the simulation, the explorer and the tanker agents coincide within the same communication range, marking the commencement of communication between the two agents. This communication unfolds through the following sequence of steps: First, the explorer agent transmits a request to the tanker, inquiring about its identification and current location on the map. Subsequently, the tanker responds to the explorer's query, furnishing the required information. Leveraging its knowledge of the map and treasure locations, the explorer then provides the tanker with the optimal path to reach the nearby targeted treasure.

4.4.3 Final Implementation

This algorithm outlines the behaviors of a Tanker Agent during its mission. Within a continuous loop, the Tanker engages in a random walk, navigating through different nodes on the map. When the Tanker encounters an Explorer, it initiates communication to solicit instructions on where to move next. Following the instructions received, the Tanker moves to the designated destination node. Upon reaching the destination, the Tanker assumes a stationary position, awaiting the arrival of treasures from Collector Agents. The Tanker remains in this position until the conclusion of the simulation, marking the completion of its mission.

4.5 Summary

In the enhanced implementation, the system has been strategically designed to operate in a distributed fashion, thereby enabling parallel processing and enhancing scalability. The Explorer agents retain their role in generating map knowledge, meticulously recording visited nodes and discovered treasures. Moreover, they engage in communication with other Explorer agents, facilitating the transmission of the map. Once the map is completely explored, Explorer agents initiate a random walk and engage with Collector and Tanker agents in a distributed manner. This involves furnishing them with information regarding treasure locations and optimal routes for retrieval. Collector agents are entrusted with the task of collecting and transporting treasures, while Tanker agents focus on covering extensive areas to expedite treasure delivery to collectors. The distributed architecture not only enhances system performance but also augments fault tolerance, enabling the system to continue functioning even if certain agents encounter issues over time. Collectively, the

agents collaborate in a distributed fashion to efficiently explore, gather, and transport treasures throughout the map.

5 Testing and Challenges

5.1 Randomness

While it is theoretically possible for all agents to eventually collect all treasures by moving randomly across a map, this approach becomes inefficient and time-consuming, particularly with larger maps such as our initial map. Despite observing faster solutions in smaller grids, the random movement strategy may take hours to unlock, collect, and deliver treasures in more extensive environments. To address this challenge, we introduced a system of treasure tracking and mission requests. Collectors maintain and share information about the location and status of remaining treasures. Lacking map knowledge, collectors, when moving randomly and encountering an explorer, request the shortest path to the nearest accessible treasure of the same type and with a sufficient quantity to collect. The explorer computes potential paths and, if available, relays them back to the collector.

Upon receiving this information, the collector transitions into a mission state, deviating from random movement. It follows a predefined path, resuming random behavior only after successfully reaching the goal and collecting the designated treasure. This mechanism proves especially beneficial in advanced simulation stages when explorers possess comprehensive map information, collectors are aware of all remaining treasures, and random movement proves inefficient in obtaining the final treasures. Ultimately, this approach accelerates the simulation process.

Furthermore, to make sure that our agents did not get stuck going back and forth between two different nodes, we introduced a buffer mechanism that tracks the last n visited nodes, aiming to avoid revisiting them. Consequently, if an agent finds itself on a lengthy path, it is constrained to move only forward or backward. Given that the nodes behind the agent have already been visited, backtracking is not possible, compelling the agent to prioritize forward movement toward unvisited nodes.

While this approach minimizes potential blockages between agents, instances may arise where the sole unvisited node is occupied by another agent. In such cases, the agent will attempt to move to any available node, regardless of its visited status, and reset its buffer to enable exploration in a new direction.

This enhancement has resulted in a more fluid and organic movement for all agents employing random movement, enabling them to navigate the map seamlessly without encountering issues or getting stuck.

5.2 Explorer's Random Walk

The issue of random movement has been resolved, but a new concern arises during the exploration phase where explorers deviate from fixed paths and instead pursue the shortest route from their current node to a goal node. A conflict arises when two explorers intersect paths, as one may seek to reach a node where the other intends to move in the opposite direction. To address this, leveraging the explorers' map knowledge, a simple solution was devised: intermittently reintroduce the random walk behavior for a limited number of steps, transitioning back to exploration afterward. This mechanism is triggered when an explorer remains stuck for consecutive cycles. While it does

not guarantee a resolution every time, testing revealed that explorers rarely experienced prolonged periods of being stuck. This modification transformed the explorers from a state of perpetual exploration without completion to a more flexible approach, allowing them to conclude exploration more reliably.

5.3 Agent Detection

We encountered a challenge in managing the interaction between agents, specifically in their ability to detect nearby agents for interaction. Due to the initial problem constraints, the agents' detection radius is initially set to zero, necessitating the continuous sending of messages in search of specific responses to initiate more meaningful interactions.

This continuous messaging requires vigilant agents who are constantly alert to intercept and respond to incoming messages. Without proper protocols or identified numbers, i.e., without precise addressing, messages may reach unintended recipients, leading to incorrect responses and causing disruptions in the overall simulation.

To address this issue, we leverage the messaging tools provided by Jade. By configuring these tools with specific protocols based on the type of interaction to be conducted, we significantly enhance the communication flow between agents. This approach facilitates faster and more efficient coordination of interactions, mitigating the challenges posed by continuous messaging and ensuring accurate and timely communication between agents.

5.4 Collector Path Intersection

Similar to the challenges encountered by explorers, collectors may face the issue of paths intersecting in opposite directions, hindering their movement. This challenge is more complex for collectors, as they lack map knowledge and cannot resort to a random walk, risking deviation too far from their mission path.

To address this, a solution involving backing off was implemented. When two collectors on a mission collide, they initiate the exchange of specific messages. Each collector receives information about the remaining portion of the other's mission path. The collector with the longer path voluntarily backs off, enabling the other agent to proceed. In cases where path lengths are equal, the collector with the higher value of their name ID takes a step back.

The backing-off process involves retracing the same nodes of the mission path until reaching a reachable node outside the other agent's mission path. After moving to this node, the agent waits for the other to progress before resuming their mission path.

With this solution, potential conflicts in agent movement have been resolved, ensuring a simulation that will eventually conclude regardless of any conflicts. Given the inherent randomness, the time to complete the map may vary in each run. Further improvements may involve enhancing the reasoning capabilities of agents or introducing new agents, to enhance overall performance without overly complicating movement in the map space.

5.5 Communication Range Experimentation

In this experiment conducted within a Dedale map comprising 500 nodes, the impact of communication ranges on three distinct agent types—explorers, collectors, and tankers—was explored using ranges of [1, 10, 50, 100, 300, 500]. The study aimed to understand how increased communication range influences agent behavior and mission success. As the communication range expands, explorers benefit from enhanced coordination and mapping efficiency; however, diminishing returns are observed due to potential redundancy and slowed decision-making. Collectors, on the other hand, experience more systematic treasure collection as longer ranges allow for better information sharing. The experiment highlighted a trade-off between communication range and performance, emphasizing the need for a balanced approach to optimize agent collaboration while minimizing communication overhead. Further investigations may explore adaptive communication strategies to address dynamic environmental changes and improve agent efficiency in complex scenarios.

In the base implementation, where the communication range was set to 100, as indicated by the problem statement, the total average execution time for our system lies at approximately 6.10 minutes. The following table demonstrates the average running times on 5 executions with each communication range:

Communication Range	Time (minutes)
1	10
10	8
50	6.75
100	6.10
300	6
500	5.25

6 Conclusions

In this project, we have created an Intelligent Multi-Agent System to address the challenge of a treasure hunt. The system involves multiple agents positioned across the map, necessitating collaborative efforts to efficiently gather all treasures. The project unfolded in several phases, beginning with the definition of the environment, agent architectures, and system structure, along with specifying agent properties. Subsequently, we explored various coordination and negotiation strategies to facilitate effective communication between the agents. Adjustments were made during the implementation phase, including simplifying the problem to fewer agent types, eliminating coordinator agents and managers, and modifying communication strategies among the agents. The outcome has proven satisfactory, demonstrating the agents' ability to intelligently coordinate and solve the treasure-hunting problem collaboratively.

Although we haven't been able to accomplish everything we hoped we would during this development, we have had to simplify the system significantly. There has been a very big learning curve for this project. However, the biggest lesson in this project has been learning how important efficiency is for multi-agent systems. These agents have to constantly be aware of where they are and the changes in their surroundings. Any big computations that put this characteristic at risk make the system harder to interact with and should therefore be avoided.

References

[1] Glavic, Mevludin (2006). *Agents and Multi-Agent Systems: A Short Introduction for Power Engineers.* Retrieved 17:36, October 16, 2020, from `https://people.montefiore.uliege.be/glavic/MAS-Intro_Tech_report.pdf`.

[2] Rocha, Jorge (2017, 28 June). *Introductory Chapter: Multi-Agent Systems.* Retrieved 21:05, October 16, 2020, from `https://www.intechopen.com/books/multi-agent-systems/introductory-chapter-multi-agent-systems`.

[3] Boissier, Olivier. *An Introduction to Multi-Agent Systems.* Retrieved 20:05, October 16, 2020, from `https://www.intechopen.com/books/multi-agent-systems/introductory-chapter-multi-agent-systems`.