**Part 3**

**Arhip Alin Gabriel**

**Multi-Agent Systems Project**

**Artificial Intelligence Master Program, 1st year**

**Introduction and problem presentation:**

Variant of the Mars Explorer scenario presented by L. Steels, where each search agent is a reactive agent. This means the agents cannot communicate between each other. They cannot store more than 1 quantity of precious rock at any given time and do not have the map of the planet. They can only know where the base is, based on the signal strength. They can also detect precious rock at close range, avoid obstacles or other agents, load the rock, once it is found and unload it when returned to base. **Implementation using Java language.**

**Design decisions and implementation solutions for Part 1:**

The map is represented as a 10x10 matrix where each cell can have the following values:

* + “o” – an empty cell.
  + “#” – an obstacle, generated random on the map, at least 1, at most 10.
  + “\*” – a raw precious rock, generated random on the map, at least 1, at most 10\*10/3.
  + “b” – a dug precious rock, transformed from raw state when a search/reactive discovers and dig it.
  + “B” the base – always situated at the center of the map.
  + “R”, “X”, “Y”, “Z”, etc. - any other upper letter indicates a search/reactive agent.
  + “r”, “x”, “y”, “z”, etc. – any other lower letter indicates the corresponding carrier/cognitive agent.

The obstacles and rocks are generated random. The initial coordinates of the agents will be near the base. The game start with each agent searching for rocks. Once an agent finds a rock, it must return to the base. Agent cannot collide one another, regardless if it has found rock or not. Once an agent has found a rock, it also cannot collide with another rock found on the way returning to the base. As such, an agent will avoid obstacles and other agents while searching for rocks and once it has found one, it will also avoid further found rocks while returning to the base.

The only place the agents can intersect is at the base. The base cannot disappear from the map, but can have more than one agent inside it at any given time. Once an agent has arrived to the base, he will disappear from the map, enter the base, unload the rock, and then start random searching for another rock, reappearing on the map. If more agents block another agent, the last one will remain still, until it has a clear path to move, regardless if it is searching for a rock, or returning to base to unload a rock.

The game ends when the last rock has been found. The base cannot communicate with the agents so they cannot know when there are no more rocks left on the map. The number of rocks on the map are always displayed, along with the movement of each agent, and their status.

**Design decisions and implementation solutions for Part 2:**

Initially, the carrier agents stay inside the base, until some rocks are discovered while the search agents explore the map. Each carrier agent cannot collide with each other or with the search agents. Also, the carrier agents cannot pass by raw precious rock, because it is considered an obstacle, just like a regular obstacle. Since the carrier cannot collect undiscovered rocks, but the user should still be able to see where the rocks are on the map, I use “\*” to represent raw/undiscovered rocks (that have not yet been found by any search agents), while “b” means the rock has been dug/discovered by a search agent and its corresponding carrier agent has been notified and it’s on its way to collect it.

The protocol is very simple: when a search agent finds rocks, it dug them and notifies its corresponding carrier by simply sending its current/rock coordinates to the carrier, at that time. Since carriers are cognitive agents, they have a map so basically they know where the rocks are located, and pretty much how to get there. When the carrier collect all rocks discovered, it will return to base and wait for the search agent to discover other rocks.

When no more undiscovered/raw rocks or dug rocks are on the map, the game ends. If only dug rocks remain on the map, the base signals the search agents to return, while the carriers go collect the last dug rocks.

**Reports and interpretations for Part1:**

The efficiency of the implementation is impacted by the number of agents. As such, more agents can gather faster more rocks versus time. I tested with 1,2,4,6, 8 and 10 agents and found that 8 is the most efficient number of search agents to gather rocks versus time. The number 8 is based on the fact that each agent can and probably will follow one of the 8 cardinal directions on the map. If more agents are added, they will block each other based on the next scenario: if two agents are going in the same cardinal direction and one is further, probably at a moment in time the last one will find a rock, and when it will try to return to the base to unload it, it will be blocked by the other agent who has traveling in the same direction.

**Reports and interpretations for Part2:**

Since the capacity of the carrier agents is unlimited, I decided to give each search agent a carrier agent. I have tested with 1, 2, 4 and 8 carrier agents, and found that 4 is the most efficient number of cognitive agents versus time, along with 4 search agents. So the optimal ratio should be 1:1.

Since they aren’t exploring the map, the carriers stay at the base while the search agents explore the map. When any rock is found, the agent that finds it will notify its carrier to come and collect it. Any successive discovery notifies the carrier in a queue manner. The carrier will collect the discovered rocks in the first-discovered-first-collected manner. Still, if in its path it happens to pass by other dug rocks, the carrier will gather it. So it is possible for a carrier to steal dug rock from another carrier, but only if it happens to be in its path. Otherwise each carrier will collect its corresponding search agent’s discovered rock.

When the carrier collects all rocks in its queue, unless the search agent discovers more rocks, the carrier will return to base to unload them. If the search agent discovers new rocks while the carrier is returning to base, the carrier will postpone the return and go collect the newly found rock.

**Proposals for improvement:**

I experimented with one carrier for all search agents but it was taking too much time to collect rocks from all around the map. More carriers were needed, but not too much though. With too much carriers, they either tend to collide with each other while gathering rocks, or staying too much in the base.

In order to decrease the time a carrier stays at the base, an improvement is to assign a carrier to more than one search agent. Unfortunately, I haven’t managed yet to found a solution for this matter.

Another improvement could be when a search agent discovers new rock to notify the closest carrier possible to its location, instead of notifying its current carrier which could be far, at the base in the worst case.