AI TECHNIQUES

MULTI-AGENT SYSTEM
STARCRAFT ASSIGNMENT

February 3, 2017

4620844 Agathe Balayn
4618092 Charu Agrawal
Group 21

Contents

Introduction

When we wrote the first report stating our strategy, we were aiming at simply fulfilling the requirements mentioned in the assignment and later, implement additional strategies in order to explore the game environment and attack the enemies. For that, we had decided on a mix between centralized and decentralized systems with all the units communicating with one agent, the Overmind, but certain units also deciding their goals by themselves. We decided so because it is the simplest way to define a strategy for a multi-agent system. Indeed, the Overmind enables us to pass on the important knowledge from an agent to another agent. For instance the drones need to know the location of the hatchery and the Overmind sends them this information that he got from the Hatchery, otherwise, hatcheries would have to send the information to all drones, and new drones would have to find a knowledged drone to get this information.

The Overmind here acts as human player who has all knowledge to himself about the environment and agents (except about the enemy), who can look at the global situation of the game and take a best decision.

Therefore, we did a first mixed implementation that works very well as not only we fulfill the requirements but we manage to build more units than expected. We also implemented an additional offensive strategy using the Zerglings. The Overmind in Starcraft can not be killed so there is no danger that we lose the unique control entity of our system. However, in other multi-agent systems, it is not reasonable to have a centralized system as the absence of the manager unit would prevent the whole system to function. That is why in a second time, we worked on a way to implement a distributed system by replacing orders from the Overmind by communication within each type of agents.

In the following sections, we first explain the centralized system and then the decentralized one that we derived from the first implementation.

1 The centralized system

We describe in this section the decision-making process for each type of agents.

1.1 Belief updates

One file GenericPercepts.mod2g is called by each entity in their event managing code. It updates their beliefs based on the percepts that each entity sees. It comprises the handling of the number of minerals, vespene and supply available to use. It also updates the amount of each type of units there are in the MAS system. Finally, it updates the isMorphing beliefs telling whether one unit is currently morphing into another one.

The main belief we use is the unitAmount which enables to keep track of the number of each type of friendly entities in the game. We use it to decide the new goals of building entities.

All entities also have a separate additional belief updating process in the event file.

1.2 Creation of units by the Hatchery

The hatchery is responsible for morphing Larvae to new entities. Its decisions are taken by itself, without communication with the Overmind. We had said that there will be communication between Hatcheries not to build too many Overlords. However, we discovered that it is better that all hatcheries build new Overlords when there is a lack of supply, because the supply need increases exponentially with the number of hatcheries. Thus, we didn't keep the communication between Hatcheries.

When there are enough resources and supplies for each entity, we first fulfill the goal of constructing Drones, then Overlord and then Zerglings in the main file, so that we always prioritize the production of drones.

Overlord At the beginning of the game, we aim at building many units, what requires many supply. Therefore, we start by building a second Overlord as there is already one in the game. Later, when our MAS becomes short on supply (it has less than 6 supplies available), each hatchery adopts the goal of building a new overlord, with the special condition that a Spawning Pool should be present on the game.

Zerglings and Drones The Zerglings are necessary to defend the system and Drones to develop the MAS. That is why we consider we need to build many of them.

The goals are adopted in the event managing file by comparing the total numbers of each entity types. When there are less than 4 Drones or 3 Zerglings, we adopt the goals of creating them. We consider these numbers as minimum requirements to play the game. Later, we adopt the construction goals in a way that we have as much drones as Zerglings. Building Zerglings also requires the presence of a Spawning Pool.

1.3 Management of the Drone actions

1.3.1 Organization of the Drones

Drones are the essential units for our requirements as they can harvest mineral and vespene gas which are the main resources to build new entities, and they can build new buildings. In this part comes the question of the centralized system. Indeed, to manage the actions of the drones, we consider that we have two types of them: the default ones which define goals by themselves, and the others which take specific actions when they are ordered by the Overmind and otherwise define their own goals. This enables us to define easily which drone is doing which specific action, and to ensure that there is always one drone devoted to this action.

The specific actions we consider are the gathering of vespene, and the constructions of the Extractor, the Spawning Pool and the Hatcheries. Drones taking care of the construction actions are not controlled anymore once they start to morph whereas the vespene gatherer keeps being considered as a drone. Therefore, we differentiate in the code between the vespene gatherer and all the other drones: the construction drones are indeed performing as normal drones until they start fulfilling their goal of constructing. Thus, in the event file, we define belief-goal rules for the vespene gatherer separately from the others.

A drone ordered by an Overmind is simply a drone which has received a specific message from the Overmind. This message results in the adoption of a specific belief by the drone. This belief is a predicate with zero variable, which is specific to the action to perform (for example, the predicate "buildSpawn/0").

1.3.2 Definition of the goals

Additionally to the general percepts, we update in the event loop their beliefs on the positions of minerals, vespene geyser, and the construction sites based on their percepts. We also update the list of each worker ID and activity, as well as the condition of the specific agent running the file.

To build all the entities we want in the MAS, we only need minerals, thus we allocate regular drones to the gathering of minerals when they are not doing anything. For that, we define a predicate "busy" which returns true in case the agent is doing an action different than waiting. When a drone is building something, we had decided that the drone would stop and start gathering minerals in case there are not enough minerals to keep building. However, we discovered that once the drone has started building, all the resources are directly allocated to the construction, and thus, there is never a lack during construction.

In case the specific drones haven't already adopted the goal of building, they adopt it under specific conditions. We aim at building one Zerg Extractor and one Spawning Pool in total, therefore the goal is adopted only when there is no such construction in the game. We decide to build hatcheries when there are enough resources and we consider that we have already produced enough Zerglings and Drones. In order to do so, we compute the addition of the current number of drones and Zerglings in the game, that we divide by 4. When the results is zero, the goal is adopted.

At the beginning of the game, we focus on the gathering of minerals, and when we consider we have enough (the number of minerals is superior to 300), the Vespene gatherer drone adopts the goal of gathering Vespene.

The main file defining which action will be taken by the drones is organized as follows: the specific action conditions are verified first so that these actions are chosen as soon as possible. Otherwise, the drone gathers minerals. In the other order, as this action is almost always possible (except when there is no more minerals on the map), the specific actions would never be chosen.

1.4 The manager: the Overmind

The Overmind does not perform any action. It only takes care of communication in order to provide needed information to the agents, and to send orders to certain agents.

1.4.1 Drone ordering

The Overmind keeps track of all the alive drones in the game in order to send them orders. It also keeps track of the number of drones alive, as it could be useful to know when the system has enough agents to start building more buildings at the same time.

The Overmind asks a specific drone to build a Spawning Pool by sending it a message, and saves in its belief base which drone is ordered. When this drone is killed, it deletes this belief from the database. Another drone is attributed to this construction. We proceed in the same way to have a drone attributed to build an extractor, an hatchery, or to gather Vespene. Therefore, there is always one drone for each specific mission. We make sure that one drone is not requested to build two different types of buildings by checking that the drone Ids is not an Id for a drone to whom the Overmind has already sent a specific message.

1.4.2 Additional communications

When an Hatchery is created, it sends its Id and position to the Overmind through the initialization function. All the drones which do not know the Hatchery position send a question message to the Overmind which replies with this position, which is inserted in their belief database. We do not manage the eventuality of hatcheries disappearing because we are not able to kill any enemy so the whole system is killed when there is an attack, and we wouldn't need the information anymore.

The Overmind keeps track of all the currently alive drones, by having in its belief base their names and Ids. When a drone is created, it sends a message to the Overmind specifying its Id so that the Overmind can update its list. When a drone dies, the Hatchery receives a percept with the Id of this drone. It sends it to the Overmind which can update its list.

1.5 The Overlords

For our requirements, the Overlords are useful only to gain supply units (8 per Overlord). Therefore, they don't need to do anything and we don't define any action to do. They would once research and upgrades are done. This however is out of scope of the project.

1.6 Additional improvements

1.6.1 Zerglings

Since we have acquired the basic requirements, it is feasible to implement a defensive strategy. We deploy the Zerglings, the most basic fighting unit into a group to form an army. Then we propose some possible useful upgrades consist in mutations of Zerglings into more powerful units.

Zerglings attack the enemy at its base location when they are of sufficient number. It is ordered by the Overmind as it keeps a count of the number of Zerglings alive. This attack is initiated as soon as more than a certain number of Zerglings are formed. For a test case we assume an amount of 8 Zerglings is sufficient to be called a strong army. The Zerglings line up at a group location within a radius of $10\sqrt{2}$ from enemy base station. They do not stand in the enemies' headquarters but some distance from them so that they do not get killed before they start killing. Since any amount of Zerglings will always be outnumbered by enemy creatures at their very base. It would be impractical to send the relatively weak army to the centre of enemy headquarters. Slowly as the number of Zerglings increase with time, the enemy notices us comes out to confront us. The enemy is exploring, so he doesn't bring along the entire fleet of army with it, but some soldiers at a time. It however gives us the opportunity to defeat fewer at time and we outnumbered the enemy. Also, we initiate this attack strategy as soon as basic creatures are created and adequate infrastructure is built. Henceforth, we do not give enemy the time to upgrade to a much advanced stage which in turn benefits us. For us to reach that stage we need to build more buildings especially research centre and invest in research.

This approach helps the agent to be defensive at the same time when it is acquiring basic requirements, without compromising on the supply of minerals and gas gathering.

1.6.2 Research Unit

Evolution chamber is the research centre we propose to build first of all the research centres because it requires minimal assets to be built. Evolution chamber allows the research of Melee attacks in Zerglings. Melee attack strengthens the Zerglings to attack with more power. The purpose is to strengthen the battle against enemy by improving each attack effect. More result in lesser time and efforts are achievable. This improves the quality of each attack by each Zerglings which proves the worth of the investment in Evolution Chamber.

1.7 Achievements with this system

At the end of one game, before the enemies attack the MAS, we manage to build: one vespene extractor, one spawning pool, two hatcheries, one Evolution Chamber, five Overlords, around 15 Drones, and around 15 Zerglings. The Zerglings form a group and attack, however they are always defeated.

2 The distribution of the initial system

In order to have a more robust system, the aim is to avoid all communications with the Overmind and redistribute all these previous communications directly between the agents of the MAS.

2.1 Redistribution of the hatchery position

To avoid communicating the hatchery positions to the Overmind, knowing that only the drones need this information, we decide to have the hatchery send its position via a message to all the drones. The drones receiving the message have to insert this knowledge in their belief database. New drones who also need this information will send a question to a random drone each time it goes through the event loop as long as they don't have this information. Otherwise, the message could be send to a drone which does not have the information, and the first drone will never get it.

However, the GOAL environment does not give the possibility to directly send messages to all agents or random agents of one type. Therefore, as it was done in the previous part for the Overmind, each drone will keep track of a list of all the current living drones of the game. Messages from the hatchery will be sent to all agents but only the agents of the right type will process them. This strategy is costly in messaging but we didn't find a way to send messages only to drones. Thus, to create the list, all drones who are initialized send a message to all agents. All agents replies by sending their own ID. The messages are processed by all the drones. The hatchery also sends messages announcing drones death to all agents.

When a new hatchery is created, it sends a message to all agents and all the drones process this. A new drone will ask the drones in its list for the information. The list enables to reduce the space of the research possibilities (as we don't ask any random entity).

The danger of this strategy is that we have to assume that there is always at least one agent who possesses an information, otherwise some information can be lost unless they are resent to everybody.

Here, we manage to get the same performances as in the implementation proposed above.

2.2 Redistribution of the drone ordering

This step is more complex. One drone has to take the decision to adopt a specific action only when no other one has taken it. For that, he sends a questions to all the other drones. One drone which is already performing the specific action will reply to the first drone and the action will not be taken. However, if there is no reply after 10 loop iterations, the action will be taken. We have to make sure here that one drone does not undertake several specific actions.

Sadly, this method is not working as it seems that all the drones are evolving at the same time, and therefore, they all are updated in the same way at the same time (at the tenth loop) and all become specific action drones, with the same action, which blocks all the future actions. The solution we used is to add a random element. Each drone belief about the specific action is updated after a random number of loops, different for each drone. With this method, we hope that drones have time to receive the message that there is another specific drone for a particular action before it itself decides to become this specific drone. However, this method is not very robust as it sometimes enables to build the different buildings but it often results in a blocked situation where all the drones try to do the same action which is finally not performed.

Attached to this report, we give two codes, one for the first mixed implementation and one for this distributed algorithm. However, we comment the trials for the drone ordering part and keep the Overmind orders for this as the solutions we have tried are not always giving satisfying results.

3 Prospective Strategy

After obtaining the minimal requirements and planning a basic attack strategy, one aims to build and ploy a strong army. First immediate extension might involve the formation of Lair. Lair is a Zerg building that unlocks many useful tech buildings and upgrades. It evolves from a Hatchery and requires the presence of spawning pool and unlocks the option of Hatchery to mutate into Hydralisk Den. Hydralisk Den creates Hydralisk which have a much better attack compared basic Zerglings. Having Hydralisk on our part improves the strength of our army. The basic steps to follow to achieve Hrdralisk involve:

1. Mutate a Hatchery into Lair
2. Create a Hydralisk Den
3. Create Hydralisk
4. Include Hydralisk in the army

This strategy might not be able to defeat the mighty enemy, but however would be a step forward to achieve a comparable army.