IMAS Final Implementation

Pau Bramon, Rachel Rapp, Björn Wahle, Carolin Wuerich

February 1, 2018

Abstract

The main goal of this work is to put in practice the basic concepts of agent technology using a real framework (JADE). To do so, the work is based on a simulation of a Metal digging environment, where different agents with different abilities are created in order to explore and obtain as many metals as possible from the system.
Keywords: Multi agent Systems, Jade, FIPA, Contract Net, MAS Simulation

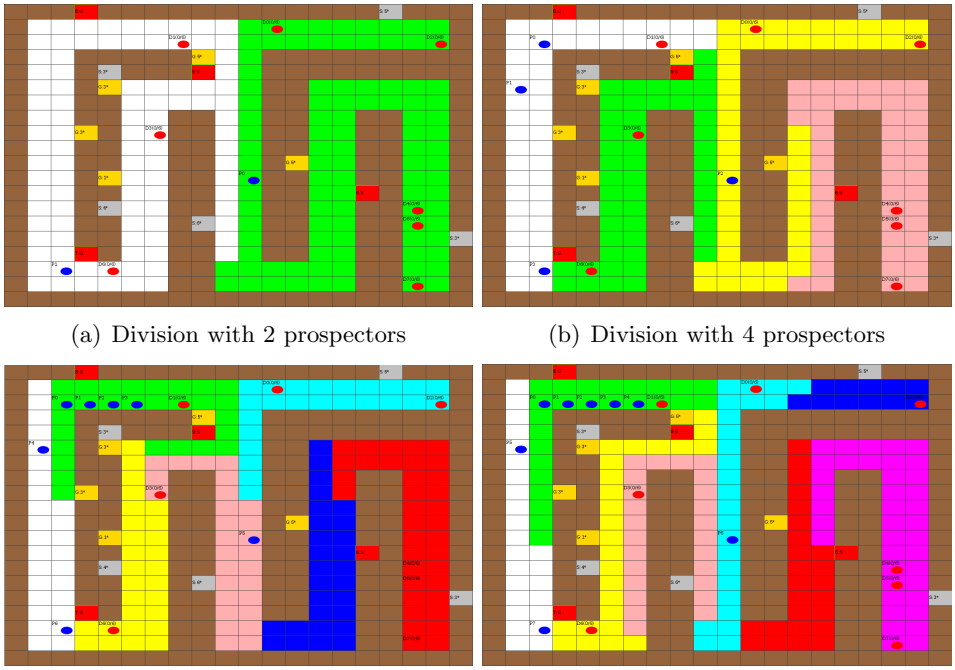
1 Implementation

In general, we did execute our plan (given in the previous submission) for the actual implementation, with only slight changes outlined below (Section 2). In the final implementation of the metal manufacturing simulation we implemented two cooperation mechanisms; the prospector agents employ implicit cooperation for the assignment of sub-areas of the map, and the digger agents and the digger coordinator cooperate via coalitions and contract net to distribute the mining tasks.

1.1 Prospectors: Implicit Coordination

Initially, the list of sub-areas to observe is empty. The Prospector Coordinator (PC) then separates the map into as many sub-areas as there are prospectors in the system. This division is approximate, the algorithm tries to make this sub-areas as equally-sized as possible, but in order to ensure that the areas are compact, they may be slightly different. The algorithm ensures that all cells are assigned to one group, but it does not guarantee

that all groups are equally sized. Figure 1 shows some examples of this division using different number of prospectors.



(a) Division with 2 prospectors

(b) Division with 4 prospectors

(c) Division with 7 prospectors

(d) Division with 8 prospectors

Figure 1: Map distribution for different number of prospectors.

Those sub-areas are added to a synchronous list within the gameSettings object sent to all prospectors at each round. Then each prospector will decide which are the most convenient areas and it will create an ordered list of preference that will be sent to the Prospector coordinator.

Finally the prospector coordinator, will get the preference lists from the prospectors and it will assign the preferred area for each prospector from the remaining ones. Therefore, the first prospector to send the message will surely get its preferred option, but once assigned, the sub-area will be removed from the list of available sub-ares. Consequently, any agent with the same preferred option will be assigned with a less preferred option.

Once all the agents have a sub-area of the map assigned, they will go there and start exploring. Prospectors do the exploration by selecting the next move based on the number of times a cell has been explored before, remem-

bering the places where it has been before. So at each round, each prospector will move to the least explored cell from the accessible ones.

In the worst case scenario multiple agents select the same sub-area to observe as the preferred one. This problem could be solved implementing a more sophisticated mechanism like, for example, Contract Net. However, an unfair distribution of the sub-areas only has an impact in the first rounds of the simulation when the prospectors need to move to their assigned area; it will later be irrelevant as they will already be in their assigned areas.

This solution was relatively simple to implement, and did not need require any complex calculations for finding the best distribution, or communication between the agents.

1.2 Diggers: Contract Net

To distribute the mining tasks amongst the digger agents we implemented Contract Net.

Digger Coordinator Agent - DelegateTasksBehaviour: During the first stage, the Digger Coordinator agent assesses the current state and recognizes if there is metal which still need to be assigned to a digger agent. The Digger Coordinator then adds the task to the list which will include its constraints, like which metal and how many units are involved, and where is it located.

Digger Coordinator Agent -TaskContractNetInitiatorBehaviour: Announcements of new tasks are broadcasted to all of the digger agents. After all of the digger agents reply, the Digger coordinator accepts the best proposal.

Digger Agents - TaskContractNetResponderBehaviour: Each Agent decides if they want to bid for the task, depending on the type of metal and its carrying capacity, and sends a proposal or refuses.

We dont consider the manufacturing of the ore to be part of the coordination mechanisms. This is handled implicitly by each agent, based on the amount of ore they are carrying, the distance from the manufacturing centers and the price each manufacturing center pays for one unit of the metal.

Additionally we changed the strategy for giving out the tasks for the digger agents. The tasks are sorted by the distance/points ratio (distance to the next manufacturing center plus rounds to finish collection and total points

paid for the amount of metal the agent is carrying) before they are announced sequentially for the contract net. This leads to diggers working in the neighborhood of the manufacturing centers as long as there is ore available, and speeds up the collection process. As shown in table 1, the average time for collecting the metal when sorting the tasks is less than the half of the time without sorting, and we obtain an increase of more than 50% in the total benefits.

	without sorting	**sorted pointsPerRound**
Total benefits	3784	5773
Total SILVER manufactured	238	381
Total GOLD manufactured	233	332
Total metal manufactured	471	713
Average benefit per unit	8.03	8.1
Average time for discovery	35.33	35.86
Average time for collection	202.26	87.42
Total metal on map	1208	1476
Ratio of discovered metal	0.97	0.94
Ratio of collected metal	0.39	0.48

Table 1: Statistics after round 600.

2 Deviations from Planned Implementation

When implementing the real system we faced some problems with the programming environment and we could not implement everything as planned. In this section, we will explain these differences that could may be added in future works. In contrast to the definition in previous reports, we made two changes in the final implementation: the coalition formation differs a bit from the planned one and the path conflicts are just avoided.

2.1 Coalition formation

The first difference from the planned implementation is that we did not implement the coalition formation as we originally planned. We planned that the diggers would form coalitions between them autonomously, without supervision of the digger coordinator and deciding themselves whether they need it or not. Instead, we did a different and much simpler thing, the digger coordinator creates these coalitions if it is necessary and the diggers cooperate to dig the metal in an implicit way. For instance, if there

is more metal on one cell than a single digger agent is able to carry, the digger coordinator agent can decide to accept more than one proposal for the contract net, leading to more than one winner to carry out the task. Thus, two diggers share the task of digging the field, however they are not aware of the other digger agent assigned to the same task (in contrast with the structure of coalitions).

We decided not to implement the coalition formation, as it doesn't bring many advantages but increases a lot the complexity. It's not necessary for the digger agents to know about the other digger, since they don't need to communicate with each other, but they can solve the task independently. Also, the calculation of the coalition values can be very memory-consuming. The solution we implemented avoids this costly calculation and yet there can still be multiple digger agents sharing a task.

2.2 Dealing with path conflicts

The second difference in the implementation is the way the agents deal with path conflicts. This is when an agent finds a digger working in the middle of the way, it cannot go through the same cell and must wait or find another path.

Initially we wanted to solve this problems by calculating a new path, modifying a bit the planned route to avoid the conflict. However, the algorithm to calculate the optimal path is complex and very time consuming. Changing it to deal with this conflicts was complex and lead to more complex algorithms.

Instead, we decided that the agent should wait for the digging agent to finish, rather than looking for the second best path. We decided this because we saw that in most situations it is unlikely that taking the second best path is faster than waiting for the other agent to finish, and it also enables the system to avoid extra computation.

3 Future Improvements

In this section we describe the main aspects that could be improved in the future to get a better performance. These are the main weaknesses of our work and we propose some possible solutions to improve them.

3.1 Map distribution

The way the prospector coordinator divides the map and the way each sub-area is assigned to each prospector can be clearly improved. On the one hand, the algorithm to find the optimal division give relatively good results, but it is far from optimal. It is implemented using a sequential assignation of cells for each prospector, instead of doing a global division. It was done this way to reduce computational cost and complexity, since finding an optimal division in complex maps can be very time consuming and the algorithm can be really complex.

On the other hand, the way each sub-area is assigned to each prospector is clearly not the best one. We decided that this improvement was not really important, since this only affects the first steps, when the prospectors go to their assigned area. However, we can notice in the simulations than sometimes this assignment could be done better. For example, a better solution could be using a Contract Net mechanism to do it, where the prospector coordinator could judge which is the best assignment in each scenario.

3.2 Prospector exploration algorithm

So far, the way prospectors explore the map once they arrive to the assigned area is in a reactive way. They decide the next move based on what they see in each round, moving to the cell less explored so far. This sometimes leads to inefficient explorations, since the decision is only based on the neighbour cells. A better implementation that could be done in future works is making the prospectors plan the best exploration path when they arrive to the assigned sub-area. This is an optimization problem, where the goal is finding the way to visit all cells minimizing the number of steps to do it.

3.3 Conflict situations

As it was stated in Section 2, we could not implement a better way for dealing with path conflicts, when an agent finds a digger working in the middle of the way. This is also a very important point that could be improved in future works, by modifying the algorithm for finding the optimal path.

3.4 Computational performance

We can notice that the simulations slows down when we keep it running for a while. We assume the memory usage is increasing with each round, probably due to programming bad practices. The program should be debugged in order to find those mistakes that lead to this effect.

Furthermore, the algorithms to find the optimal path and the best division are far from computationally optimal. A good improvement for future works could be improving the way these were program to reduce computational time.

4 Conclusions

Overall, it was quite straightforward to implement the project as we had planned. However, we were glad to not have chosen other coordination methods, as they would have been very complicated to implement by involving many exchanges of messages or time in general to be executed (e.g. auctions and partial global planning).