Technical University of Denmark

02285 Artificial Intelligence and Multi-agent Systems

Mandatory Assignment 2

Authors:

Andreas Hallberg Kjeldsen
s092638@student.dtu.dk

Morten Chabert Eskesen
s133304@student.dtu.dk

Peter Carlslund
s113998@student.dtu.dk

April 1, 2014

April 1, 2014 | Andreas Kjeldsen (s092638)
Morten Eskesen (s133304)
Artificial Intelligence and Multi-agent systems | Peter Carlslund (s113998)

Introduction

This assignment is about being able to explore an unknown environment with multiple agents. Furthermore the agents should be able to coordinate their position such that their team achieves a good score. The work done is based upon our previous work for assignment 1. This report documents our analysis of the problem, our solution and lastly the achieved results.

Problem Analysis

We have managed to map the environment with a single agent. Now we must use multiple agents to do it. This leads to some new possible problems to overcome.

- **Faster Mapping**
  Having multiple agents at our disposal should result in the environment being mapped faster.
- **Working Together**
  To map the environment faster, the agents should work together, which means communicating information between the agents.
- **Coordinating Actions**
  There is no reason to do the same work multiple times, therefore the agents should coordinate their actions.
- **Achieving A Good Score**
  When the environment has been mapped, the agents should place themselves in strategically good places to obtain a good score.

The limitations from the first assignment still applies: No knowledge of direction, only able to see two 'spots' away.

Solution

Faster Mapping

With multiple agents at our disposal we had to decide how the mapping should be performed. We came to the conclusion, that the agents should map the things closest to them, and not try to state which parts of the environment they would handle alone etc. The argument was that the agents cant know if taking a path will reveal lots of new territory, thereby more work to be done, or no territory at all.

We started by simply adding the 7 extra explorer agents and starting the game to see how they performed. The agents were not instructed to share knowledge nor coordinate their actions. We could see that depending on where the agents were dropped, they mapped the environment at different speeds. Most of the time, all agents had mapped the environment somewhere between step 500 and 600. We noticed that if two or more agents was located at the same vertex, they tended to follow the same path when mapping the rest of the environment.

Working Together

To improve the speed of mapping the environment, the agents needed to share their knowledge of the environment. At first we kept all the information gathered decentralized, meaning each agent had their own knowledge base. We decided that when an agent perceives something new, the agent should share the new information with the other agents. The information would be exchanged between the agents using a messaging system. When an agent perceived something new, they would broadcast it via the messaging system. When the other agents had to plan their next action, the could check for new messages. The messages would be interpreted by the agent, thereby allowing the agent to include the contents of the message in its own knowledge base.

There was one problem with the messaging solution. It was always one step behind. Furthermore it did cause a minor overhead, sending, receiving and handling messages. We decided to centralize the knowledge base for the agents. Each agent would then instantly have access to the same information about the environment, there by eliminating the 'off by one' synchronization.

The agents now share their knowledge. They all know when the complete environment has been mapped. An improvement in the speed of mapping the environment was also noticed. The amount of steps required had dropped to be between 350 and 400. But we still saw the same tendency that two agents would follow the same path, thereby wasting steps.

Coordinating Actions

Next step to improving the mapping speed was to make the agents coordinate their actions. We took the centralized approach, as with the knowledge sharing. We had to come up with some sort of system, where the agents could tell what they planned to do. The system should support the following:

- An unprobed vertex should only be probed by a single agent.
- Unsurveyed edges visible at a vertex, should only be probed by a single agent.
- When looking for unprobed vertices, only the agent with the shortest path in terms of steps required, should proceed towards the unprobed vertex.

We created a system, we'll be referring to as the *planning center*. The planning center keeps track of the actions the agents wish to perform. It refuses actions that would be a duplicate of an already planned action, while also preferring actions that are more beneficial than already planned actions. A more beneficial plan could be going to an unprobed vertex in as few steps as possible. The agents will only perform their action, when all agents have an action to perform that does not conflict with the others.

The agents plan their turn in a sequential order, starting from agent `A1` to agent `A8`. Therefore we had to make the planning center, act as some sort of bidding system. The agents bid on the action they wish to perform, using how much the action would *cost* as their offer. Some actions are 'first come, first serve', these include probing of a specific vertex or surveying from a specific vertex. Other actions are given to the agent with the best offer. Best offer is determined based on the action. For going to and probing an unprobed vertex, the best offer is the lowest amount of steps required to reach the vertex.

With the planning center in place, the agents no longer have conflicting actions. The environment is now mapped even faster. The amount of steps required to map the environment is now usually between 200 and 275.

Achieving A Good Score

The agents would still be idle when the environment had been mapped. With the environment mapping speed greatly improved, the agents were spending even more time being idle and not contributing at all. We therefore wanted the agents to contribute to achieving a good score.

The game grants points to a team based on various factors, one of them is the *zone score* of the team. The problem definition purposely defined the notion of a good score vaguely. We decided that the score, would be defined as the zone score of the team. A good score would therefore be a high zone score.

To achieve a high zone score, the agents have to place themselves strategically. We wanted to find a subgraph that would result in a high zone score. We decided to split the environment up into smaller connected components of a maximal size. The maximal size would be based on the amount of agents we have access to. When the environment has been split into subgraphs of maximal size, we place the agents within the subgraph that we find to have the highest zone score.

The agents must be instructed to go to a specific vertex. This was already implemented in the first assignment, but we came up with a new heuristic to improve the path finding. Our first heuristic simply used the weight of the edges. Our new heuristic uses both the weight of the edge and the amount of steps it has to perform. This way we have a preference towards shorter paths. If the agent can reach the vertex it is instructed to go to quickly, then the zone will be enlarged faster, resulting in a better score.

Results

We are still able to map the whole environment. We have speed up the process with the help of the new agents added. The time taken to map the whole environment has been more than halved. The speed improvement has been possible due to centralizing both knowledge and action planning. Centralizing the knowledge made it redundant to pass messages between the agents. Centralizing the action planning, made it possible to avoid conflicting actions and made distribution of work possible.

Our agents can place themselves strategically to achieve a good zone score. Our algorithm for finding a good zone score could be improved. We do not consider placing agents at vertices with high degree, though it could result in having a larger zone. Further the opponent team does not move right now, therefore we do not take into consideration that the zone might be broken. When the agents place themselves, they do not move afterwards. If an agent is instructed to go somewhere else after having been placed within the zone, the agent does not automatically return to its position within the zone. This will most likely be solved when having to take the opponent team into consideration, as the best possible zone will change over time, forcing the agents to move around.