Final Report
Multi-Agent Programming Contest

Rahul Arora, Artur Daudrich, Sergey Dedukh, Michael Ruster, Michael Sewell, and Yuan Sun

University Koblenz-Landau

1 Motivation

2 Scientific Background and Fundamentals

2.1 Agent Programming Concepts

2.1.1 BDI.

2.1.2 Formal Methods.

2.1.3 Negotiation and Argumentation.

2.1.4 Agent Societies.

2.2 Agent Programming Languages

2.2.1 GOLOG and FLUX.

2.2.2 Jadex.

2.2.3 AgentSpeak (L) and Jason. Why are AS(L) and Jason so awesome that we chose them over the other options? Why didn't we invent our own language or at least our own system/architecture/infrastructure?

2.3 MAPC: Contest and Scenario

What is the general scenario? Agents on mars trying to find water in a competitive manner against another team. Explain the concept of zones, gaining points, achievements and also how agents differ from each other.

3 Team Organisation

3.1 Structure and Meetings

Dynamic working groups that were built weekly to tackle the newly crafted tasks per week. In the beginning, we also tried some hacking sessions. But we quickly found out that working from home works best for us. Plans and discussions were held together in the weekly meetings (on the whiteboard).

3.2 Git, Hangouts and Skype

Revisionsing system, Wiki for minutes and issues for problems. VoIP-solutions for collaborative programming. Not all problems were transformed into issues. Some were just mentioned and discussed in the Hangouts group chat and then quickly solved after.

4 Architectural (?) Structure

4.1 Agents

Talk a bit about generalisation e.g. a saboteur is a specialisation of an agent. I.e. both share exploring but the saboteur also knows how and when to attack. Explain what tasks our agents have and where our priorities are.

4.2 Simulation Phases

Explain the general split up into an exploration and a zoning phase. Also mention the parallel aggressive strategy of the saboteurs. Talk about repairers, explorers and inspectors and their special tasks but without going into details abouttheir complete strategy (this is part of the next chapter).

5 Algorithms and Strategies

5.1 General Strategy Overview

This could also be an introductionary text which motivates the following subsections.

5.2 DSDV

What is it? How is it used in our context? What are advantages we gain from it? What is problematic (speed loss)?

5.3 Exploration

How do agents move around during the exploration phase?

5.4 Zone Forming

Zoning happens asynchronously. Introduce the general concept and motivate our ideas here.

5.4.1 Zoning roles There are coaches and minions. Coaches command minions. There are also agents who don't get assigned a specific role and try to find and go to a well. There is a strict hierarchy between the roles.

5.4.2 Colouring algorithm for zone finding In some way, we try to find local maxima to build small zones with as few agents as possible. How do we find zones? How do we find out how many agents we need? Present our colouring algorithm.

5.4.3 Zone Extensions and Breakups Extensions aren't implemented but breakups are. Maybe merge this with our first zoning section.

5.5 Agent specific strategies

How are the agents specialised? Explorers keep probing for long. Inspectors probe enemies so that we can avoid saboteurs. Saboteurs are attacking and can be called for zone defence. Disabled agents communicate with repairers and approach them.

6 Implementation Details

6.1 BDI in AS(L) and Jason

Or in general: how did we implement what we had learnt from our scientific background?

6.2 Information Flow

Who gets what information how and when? How do we communicate with the server?

6.3 Lifecycle of one Step

Maybe illustrate what happens within one step and how we prevent multiple actions to be executed in one step.

6.4 Competition results

What place did we rank? How did the others do? Analyse our matches shortly and point out problems we faced, how we tackled them and point out what had gone well.

6.5 Lessons Learned

Here we could explain what was working well and what was troublesome. Java: fast. AS(L): slow and hard for us to program. Communication: extreme bottleneck. Also we could note that all this would need much more time and preparation (or a team that is more familiar with agent programming). We can illustrate this with our approaches of a dedicated cartographer agent and the node agents. We might also illustrate further failed approaches.

7 Conclusion