University of Southern Denmark – Course AM24

Multi-Agent Systems

Planet Exploration

Group 3

Martin Moghadam, Kalle Grafström, Morten Knudsen

12-08-2010

Table of Contents

Introduction

A project created for the summer course of AM24 – Multi-Agent Systems, using Madkit and Turtlekit, developed with Java (Java JRE6).

The project involves multiple agents for planet exploration with robots; ore is harvested from the planet and stored in bases, using explorers to find the ore, and transporters to move the ore to the base. The explorers and transporters have a limited amount of energy available, when energy is almost depleted they return to base to recharge. Energy is consumed by actions; move, send message, perceiving the environment. If a robot depletes the energy before it can recharge the robot dies. The explorers have a limited perception scope and can only detect ore that is nearby. The transporters can pick up the ore when they are at the same position. Each base has a limited capacity of ore, and each transporter can carry a limited amount of ore. Robots return to base when; the ore has filled the base, or the time has run out.

The table below shows the parameters of the project and the associated symbols which are used throughout the report.

Base Capacity	C
Ore Density	D
RobotEnergy	E
Grid Size	G
Mode	M
Number of Bases	N
RobotPerceptionScope	P
RobotCommunication Scope	I
Robot Memory Size	S
Max Simulation Time	T
Transporter Ore Capacity	W
Explorer Amount	X
Transporter Amount	Y

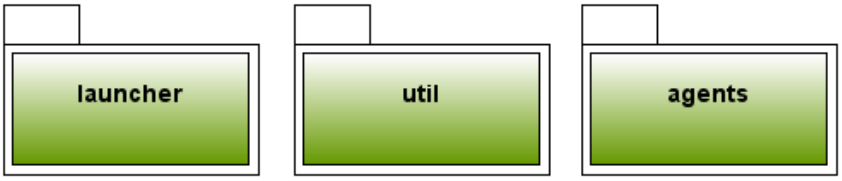
The motivation of the project was:

- Becoming familiar with Madkit and TurtleKit, understand the tools and examining the online documentation and examples.
- Learning to develop Multi-agent systems and creating communication between the agents.
- Creating smart agents that are not just reactive, but proactive and cooperative.
- Testing, experimenting and improving the agents, and examining solutions of the other groups to see the different possibilities and gain experience.

The report documents the development and experiments, then discusses the results, examines further development possibilities and draws a conclusion.

Design

The project consist of three packages; launcher, util, agents.



- The launcher package contains the initialization of the project;
  - Creating the ore as pink patches on the grid.
  - Adding the simulation agents.
- The util package contains the utility classes;
  - The constraints and parameters.
  - Queue for managing jobs.

    - Messaging system for communication between the agents.
- The agent package contains the agents;
    - The base agent.
    - The explorer agent.
    - The transporter agent.

Constraints and Parameters

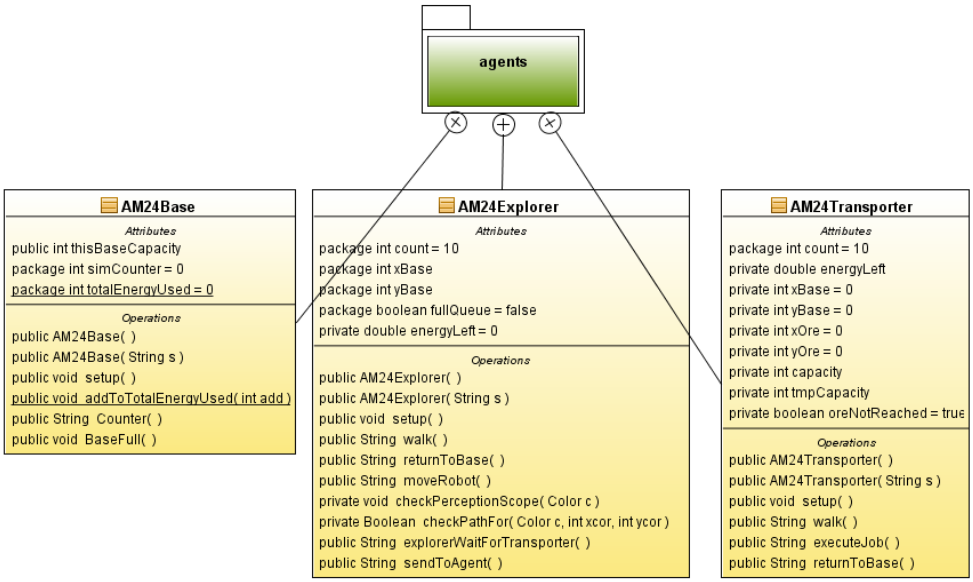
Constraints and parameters are defined in AM24Constraint class as public final fields that can be accessed by the agents.

Queues and Jobs

The Java library's Collection classes has been used to create a queue of jobs, the ArrayBlockingQueue is used to create an array list consisting of objects; of the AM24Job class. The queue stores the jobs, which are the passed between the agents as messages.

Messages

The message class contains the job which can be received with a get method.



The class diagram shows the agents and the agent package, which is examined in the following sections.

The color of ore is pink.

Base

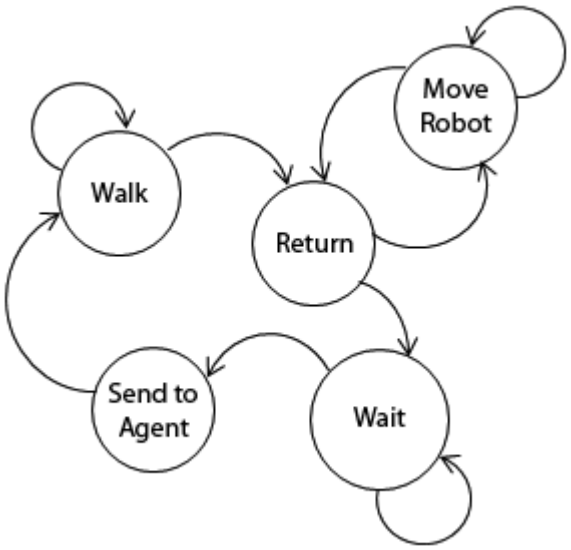
The base is Green.

The base;

- Sums up the total energy cost.
- Counts the simulation time.
- Checks if the base is full of ore, by examining the base ore capacity.

Explorer

The explorer is Red. The state diagram shows the different state of the explorer.



- Walk; the explorer walk through the environment.
    - As the explorer move it checks the perception scope for ore.
- Return; when the queue is full of job the explorer set heading to returns to base.
    - Move Robot; The explorer then to the base.
- Wait; the explorer waits for a transporter agent and then;
    - Sends the queue of job to the agent.
- The explorer then continues to walk.

Walking

The explorer moves forward in a random heading, and after moving in that direction for a period according to a counter the direction is changed to a new random heading. The pattern is a lot like are termite, and the termite demo from Turtlekit was examined for inspiration. When the explorer moves it check the perception scope for ore.

Perception

The explorer checks for ore by examining the patches within the range of perception, the explorer iterates though the different patches within its perception and checks if the color is that of the ore. When ore has been found a job is created and added to the queue for harvesting.

Warp

When warp is enabled finding ore also warp around the planet, if the explorer is located near the edge of the grid the perception can see the other side of the planet when it is within range.

Send to Transporter

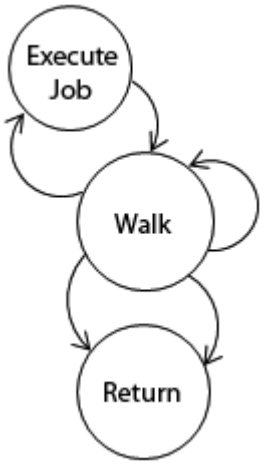
When the queue is full and the transporter is at the base; the jobs in the queue are polled and sent to a transporter agent, which is selected at random.

Energy and Time Cost

The move and check perception for ore consumes energy, the explorers is recharged at the base. Changing states consumes time.

Transporter

The transporter is blue. The state diagram shows the different states of the transporter.



Walking

The transporter polls it job queue which is a FIFO buffer and checks if it has enough energy to execute the job and reach the home base. If this function is not true it will poll another job from the list and this is done until the job list is empty. If this happens the transporter will go home and unload its ore samples. At base it will get a new assignment and recharge its batteries.

Messaging and Communication

The message system used in the transporters when it communicates with the explorers is a very basic one and is not following any standard. It basically just receives the jobs that the explorers have sent to it.

The transporter looks in the message box and adds the suggested jobs into its job queue.

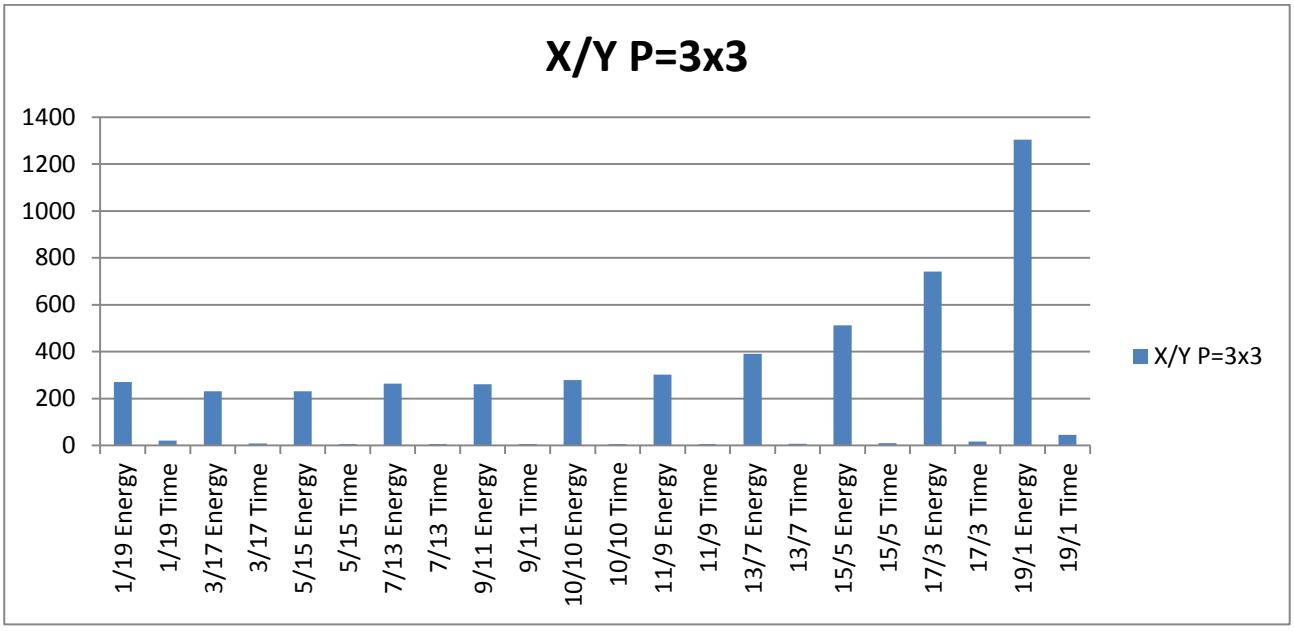
Energy and Time Cost

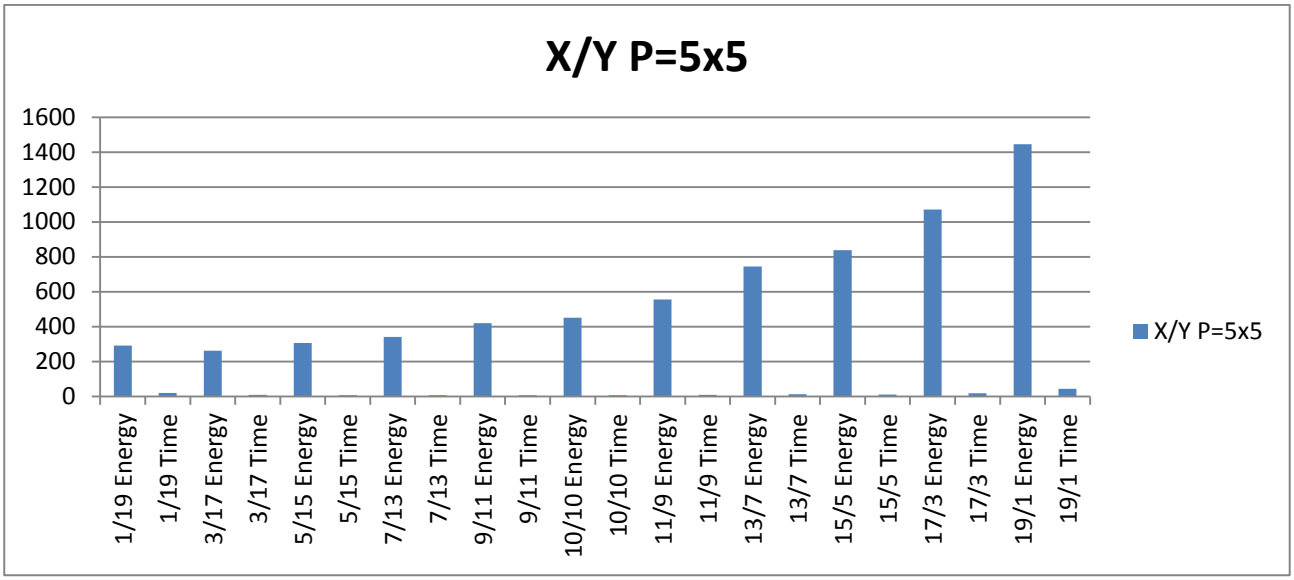
The moving is the only energy cost for the transporter, the transporters is recharged at the base. Changing states consumes time.

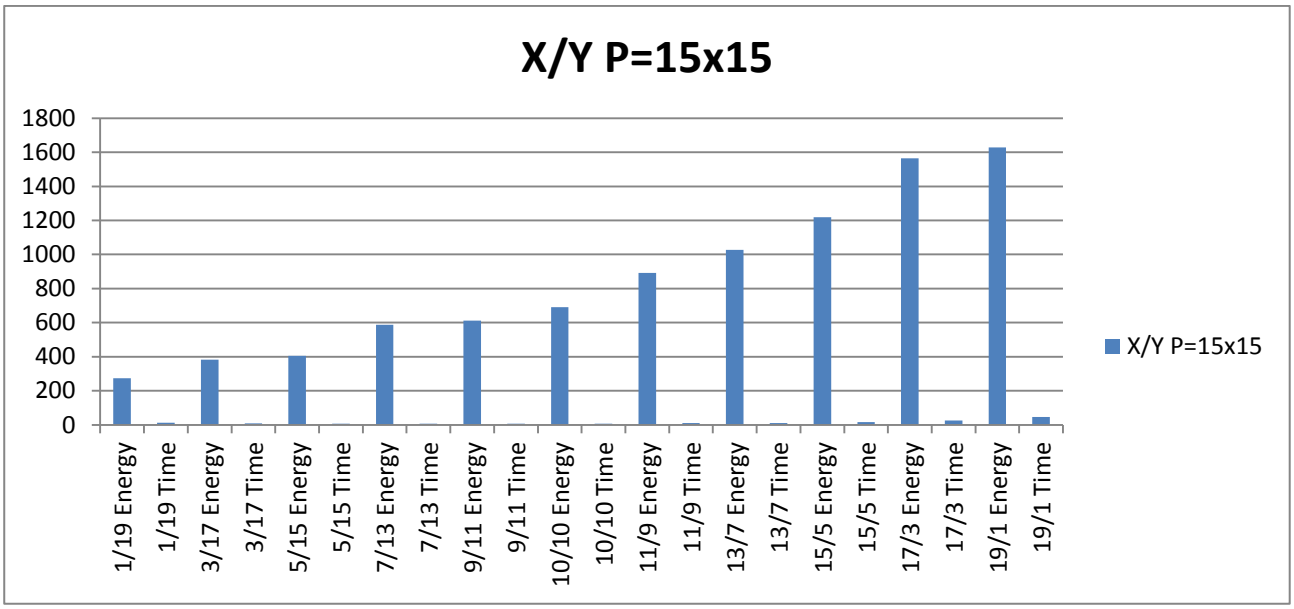
Experiments

The parameters used in the experiments where the one given by the instructor, shown in the table below:

Base Capacity	C	200
Ore Density	D	0.05 Uniform distribution
RobotEnergy	E	5000
Grid Size	G	200x200
Mode	M	Cooperative
Number of Bases	N	1
RobotPerceptionScope	P	3x3 5x5 15x15
RobotCommunication Scope	I	11x11
Robot Memory Size	S	15
Max Simulation Time	T	10000
Transporter Ore Capacity	W	8
Explorer Amount	X	10
Transporter Amount	Y	10







The experiments shows that the system consumes least energy when there are fewer explorers than transporters. The experiments are only for one base, and the parameters that were changed, were the Perception scope of the explorers and the ratio of explorers vs. transporters.

Discussion and Further Development

In the project there is a lot more development to do. Like mentioned earlier we haven't been able to implement all the functionality due to the limited time. We still need to implement cooperation mode, which is the mode where the transporters, explorers and bases works together. This means that a given

Transporter if it doesn't have a job to do, then it can go to another base to see if there are explorers there it waits for a transporter to come so that it can send the ore queue to the transporter. Furthermore we haven't implemented collision detection either; in the following sections the different we describe the intended design and optimization. Bread crumbs could also be used to optimize the exploration so that the explorers don’t add a patch that it already visited once before. It has actually been tested but were not included in the experiments.

Cooperative Mode

Like just mentioned this isn't implemented, but the idea we had where to add all the bases positions to the transporter and explorers and reducing the memory size of the job queue to the memory size minus the number and bases. This will as far as we think be an okay solution, if we have a limited number of bases.

Possible Improvements to the Logic

Explorers; Right now the Explorer moves in a random fashion, and only uses its perception scope to look for ores and nothing else. It has been our intension to make this a little more cleaver by adding an extra state to use the perception scope to point the explorer in the direction of more ores. If we look at the logic behind this idea is that it is more likely that there are more ores in the direction of the last ore found, than just doing random direction change on a counter. This could furthermore, be extended by more and more complex behaviors. Unfortunately this hasn’t been implemented so we can't compare the results to our other experiments.

Transporters; The transporters behaviors could further be improved, with a smarter shortest distance to the next ore from the transporters current position through peeking the whole memory queue and choosing the closest ore patch to go and pick up. A collision detection function would also be nice to have to mimic a real behavior that for sure would have been implemented into a real life robot. For multi base cooperative mode, an extra behavior for calculating the nearest base could improve the performance further.

Messages; The message system we use is just a simple Message system we made. It doesn't use any of the already existing message systems like ACL, ACT etc. It would be a good idea to implement one the message systems to have the functionality of bedding, offer, ACK messages and more. The reason this probably would be a little better solution is that right now we have no method of checking if the Transporter actually got all the positions of the ores from the Explorer, but in this quantitative system we just don't care, it takes all the positions it can have in its queue and just throw the rest of the queue in the "dumpster".

Conclusion

In this project we have gained valuable knowledge in multi-agent systems. Even though we haven't successfully implemented the cooperation part of the project we think that the project has been fruitful. We learned to use Madkit, Turtlekit and developing semi intelligent agents that are not very reactive, but proactive. A lot of enhancements could be included if the time constraints were not an issue.

The experiments for one base shows, that more transporters than explorers yields least energy and time used to collect the designated number of ore samples.