UNIVERSITY OF NOTTINGHAM

G54DIA COURSEWORK 2 REPORT

NAME: Sheng Wang

STUDENT ID: 4207954

EMAIL: psysw1@nottingham.ac.uk

1. Introduction

This coursework is aimed at designing multiple cooperating agents which can collect and deliver water to stations (customers) in a 2D environment. Tasks (Water Requirement) are generated by stations periodically. The environment also contains several wells which water can be collected. The expectation for this agent is to deliver as much water to as many stations as possible in a given fixed period of time.

First of all, this report will introduce the implementation of this multi- agent system from specification of the task environment, architecture design to software implementation. Then it will give analysis of the performance of this multi-agent system and illustrate the relationship between the selected architecture and the system performance.

2. Specification

Environment

The task environment given in this coursework is **inaccessible**, **deterministic**, **episodic**, **dynamic** and **discrete**.

Percept

The informations an agent can obtain in this coursework are :

About Environment :

1. **Type of a cell**
   The agent can determine the type of a cell it discovers, station, well, fuel pump or empty.
2. **Relative position**
   The agent can know the relative position of a cell based on its current position.
3. **Task**
   The agent can get the requirement of a task that a station generates as long as the station is visible.
4. **Remaining Time**
   The agent can get the current remaining timesteps from the environment.

About Agent Itself :

1. **Fuel level**
   The agent can know how much fuel is left currently.
2. **Water Level**
   The agent can know how much water is left currently.
3. **Score**
   The agent can know how much scores and tasks it has got and completed currently.

Actions

Actions avaliable for the agent are listed below :

1. **Move**
2. **Load Water**
   The agent can take water from a well.
3. **Deliver Water**
   The agent can deliver water to a station.
4. **Refuel**

The agent can refuel at the fuel pump (In this coursework, it is at the centre of the environment).

Multi-agent

This coursework aims to solve the same problem in the previous coursework with multiple cooperating agents instead. The team's performance is determined by the average amount of water delivered per agent.

One important property about this multi-agent system is that all agents can be owned only by **a single organization**, which considerably simplifies how agents interact. Details will be illustrated in later section.

3. Design

System Structure

Compared to single agent system, multi-agent system has its advantage in characterizing or designing distributed computing systems. However, in order to manage all agents to behave consistently and robustly in the multi-agent environment, two types of issues that need to be considered are proposed by multi-agent systems literature: **Coherence** and **Coordination**.

Coherence. Based on Bond and Gasser, coherence refers to "the performance of cooperation between agents in the multi-agent system". Coherence may be measured in terms of solution quality, efficiency of resource usage and conceptual clarity of operation. In this coursework, we use the average amount of water delivered per agent to measure the performance.

Coordination. Based on Bond and Gasser, coordination refers to "the degree to which the agents can avoid 'extraneous' activity". In a perfectly coordinated system, agents can achieve better the goals of themselves and of the system in which they exist.

In general, agents in a multi-agent system may have been designed and implemented by different individuals, with different goals. Therefore, they may not share common goals and have to act strategically in order to achieve better their own goal (**The Prisoner's Dilemma**). In this case, constructing a both perfectly coherent and coordinated system is nearly impossible if agents are very self-interested and there are many potential conflicts between agents (e.g. in the most extreme scenario, self-interested agents in a zero-sum encounter). And even in most realistic scenarios, using negotiation to achieve cooperation is proved to rarely be possible in practice because of the communication and computational overheads incurred (Wooldridge, 2002).

Luckily, as described in the specification, all agents of this system are owned by a single organization (me, the author), therefore, we are able to design our own **non-antagonistic** agents and use a manager agent to maintain a model of all worker agents to achieve successful cooperation.

1. Control Structure

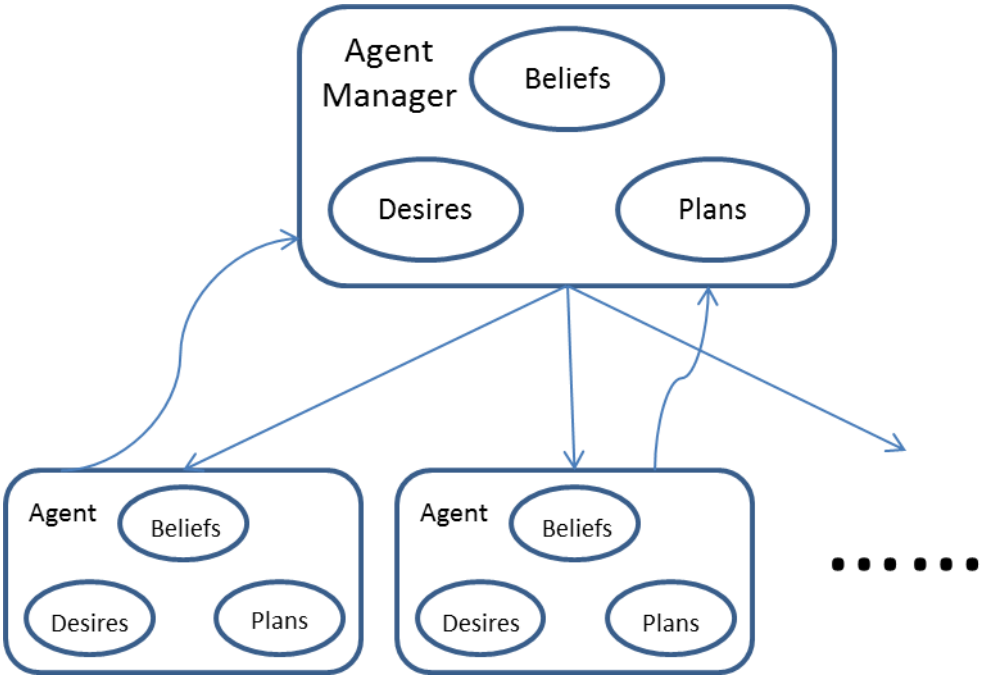

Figure 1 hierarchy of control structure

Although distributed architecture has the advantage of robustness (Component breakdown will not lead to whole system breakdown), this doesn't work for this coursework as this is a **serial program**. Besides, the complexity of distributed systems makes the development slow and difficult. Therefore, here we simply adopt **centralized control structure**, which only allows worker agents to communicate with the agent manager while the agent manager can communicate with all worker agents (see figure 1).

The agent manager in the figure 1 can be seen as a shared memory of all agents or a server (agents as clients). Compared to normal worker agents, agent manager cannot sense the environment directly. Instead the manager builds up its beliefs based on the information collected by distributed worker agents. Agent manager is responsible for **making an optimal decision** for each agent, while worker agents are responsible for **information collection** and **task execution**.

2. Agent Manager

   Architecture

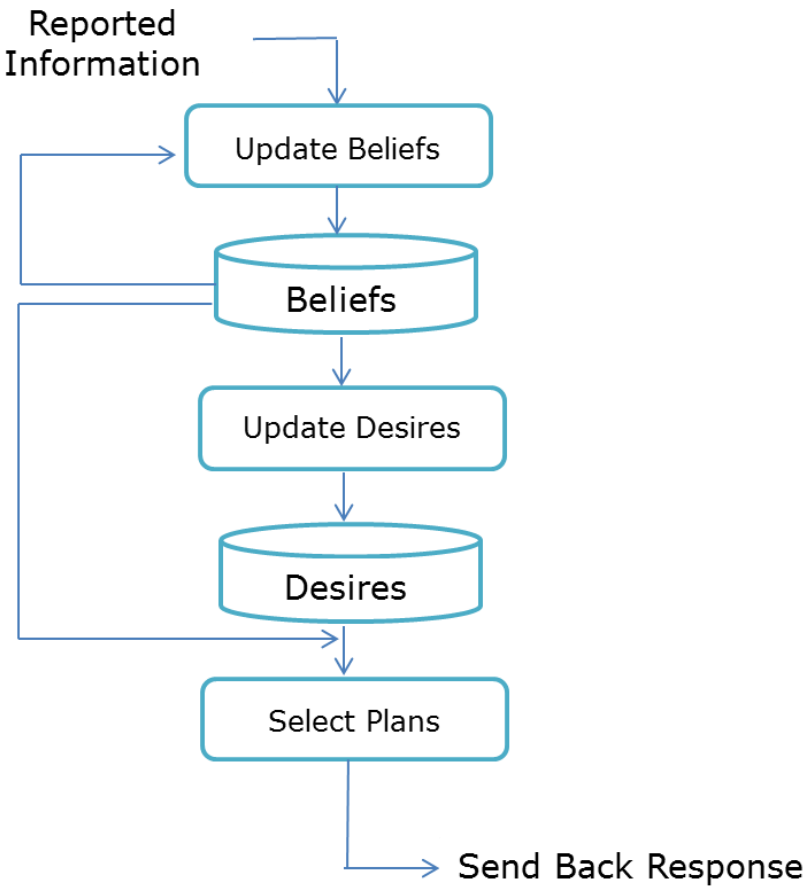

Figure 2 how agent manager updates internal states

As shown in figure 1, agent manager adopts the same "BDI" architecture as worker agents. However, as the manager cannot sense the environment (without sensors) and is only responsible for decision making (without actuators), the manager updates its beliefs by the reported information from worker agents and will return a selected plan in the form of response to agents (see figure 2).

"BDI" Model

Manager's Beliefs

Figure 3 what are included in the manager's beliefs

As shown in figure 3, what the agent manager believes can be classified into three main categories: **agent information**, **global map** and **task list** (including available tasks and allocated tasks).

Manager's Desires

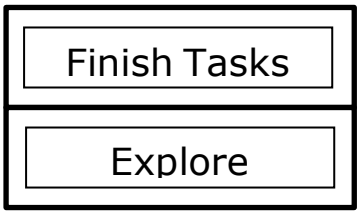

Figure 4 priority of desires in general

The **hierarchical** structure of desires used in previous single agent system is adopted for the agent manager. Therefore, if there is any task available, the agent manager will allocate tasks to every incoming request from worker agents. Otherwise, it will allocate exploration points to agents. This structure guarantees that as long as there is any detected task, the agent manager will immediately allocate tasks to worker agents.

3. Worker Agents

Type

In a multi-agent system, agents the system uses can be totipotent or specialised and these two types both have its advantages and disadvantages. In this coursework, we will choose to use **totipotent worker agents** as less specialised agents will give a more flexible and reliable system.

Architecture

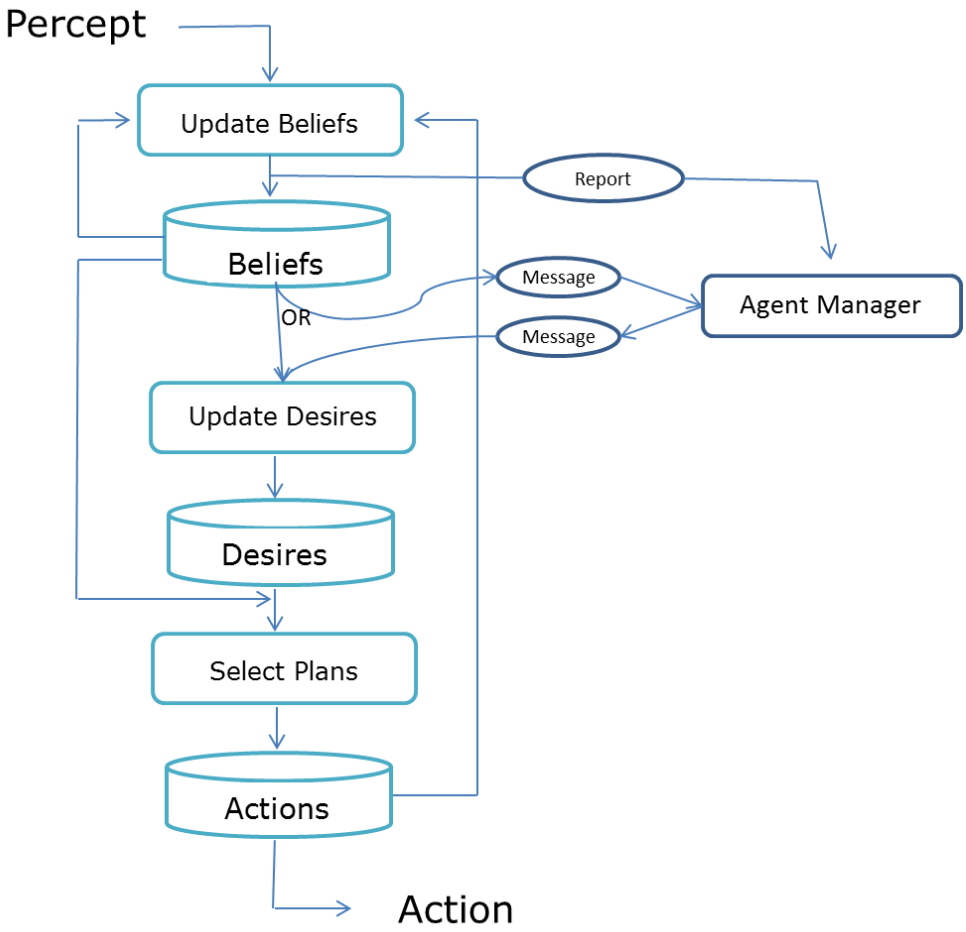

Figure 5 how worker agent updates internal states

As shown in figure 5, compared to previous single agent system, the worker agents used in this multi-agent system have nearly the same architecture. However, in order to fit in the multi-agent system, there are mainly **two modifications** that have been made to the previous single agent architecture.

Send a Report to Manager

As mentioned before, the agent manager extends single agents' observability by combining the information collected by each agent. Therefore, worker agents should be able to frequently report what they have perceived to the manager in order for both side maintaining consistent and up-to-date beliefs. The content of the report can be, for example, newly detected tasks. Besides, the manager doesn't need to respond to the report.

Send a Message to Manager

The other is that worker agents in this multi-agent system have two options after updating their own beliefs instead of

one: continue to update desires or send a message to the agent manager.

Usually every worker agent will store its current allocated task into its memory. Once the agent has finished the task, it will send a message to the manager to hand in previous allocated task and ask for a new one. Otherwise, the agent will continue to finish that task.

Although it may be more efficient to require worker agents to report current progress of allocated task every time step, this will considerably increase computational burden of the whole system. Therefore, only when they finish their current task will worker agents send task request to the manager.

"BDI" Model

Worker Agent's Beliefs

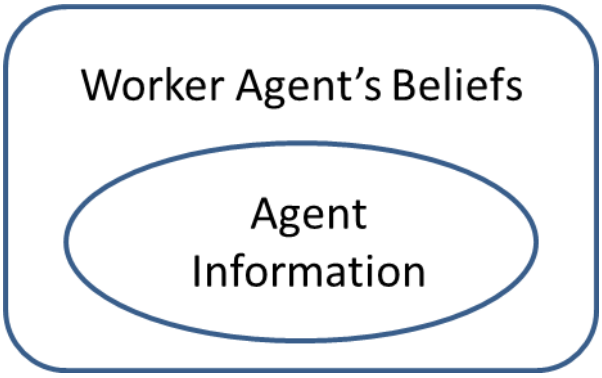

Figure 6 what is included in worker agent's beliefs

Compared to previous single agent system, beliefs of worker agents in this multi-agent system are only left with agent personal information (fuel level, water level and current position) (see figure 6). Without enough knowledge about the environment, worker agents cannot make decision and therefore are only able to follow the instruction given by the manager.

Worker Agent's Desires

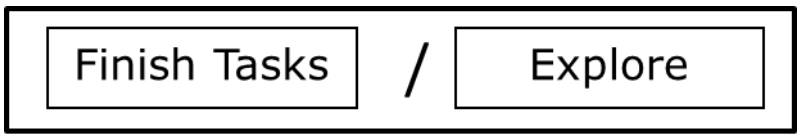

Figure 7 single layered desires of worker agents

As the agent manager takes over the work of task allocation, there is no need for worker agents to automatically switch

between "exploration" state and "doing task" state. Therefore, single layered structure (see figure 7) is adopted for worker agents' desires.

4. Communication

   Languages

   There are three kinds of messages used in this system: **report**, **request** and **response**. The first two are protocols used by worker agents while the last one is used by the agent manager. Report is used to deliver what worker agent senses to the manager, and request is used to hand in worker agent's current allocated task and ask for a new one. Response is how the agent manager allocates a task to worker agents.

   Receivers

   In order to receive those messages, both agent managers and worker agents have to implement broadcast receivers. For agent managers, "**onReceiveReport()**" and "**onReceiveRequest()**" are used to deal with "report" and "request" respectively. While for worker agents, "**onReceiveResponse()**" is used.

Problem Decomposition and Allocation

How to allocate exploration points?

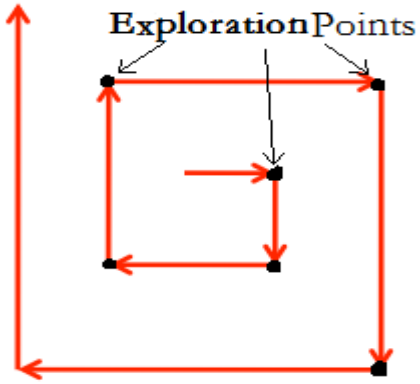

Figure 8 exploration approach selected in previous coursework

The initial exploration approach is directly adopted from previous single agent system (see figure 8).

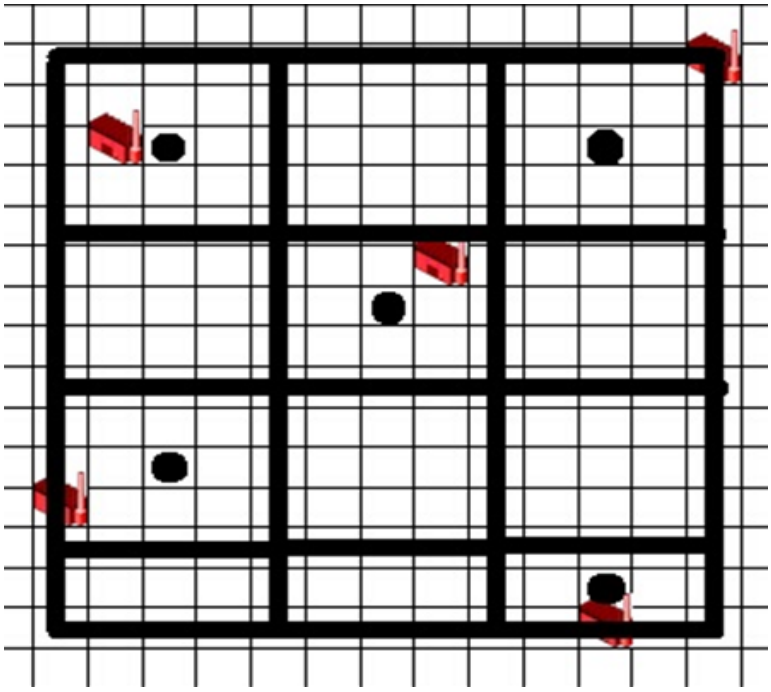

Figure 9 exploration point optimization used in previous coursework

Exploration optimization method used in this system is also the same as that in previous single agent system (see figure 9).

In terms of exploration point allocation, there are two methods tested for this system. One is allocating the **closest point** to the agent requesting for tasks. The other is allocating the point around which there is the **highest density** of stations.

However, it seems that these two methods don't make much difference in terms of performance. Here, for this system, the latter one will be adopted. Detailed test will be introduced in "Evaluation" section.

How to allocate tasks?

Allocate Task
Input: a list of tasks, agent's current position, fuel level and water level, and a table containing station-well pair where each well is the nearest well to that station

1. Test all the tasks, if there is a solution to this task that can finish the task without needing to return to the fuel pump, choose this solution
2. Otherwise, pick out a solution with the shortest path
3. Send the selected solution to the agent

As returning to the fuel pump usually has to take a lot of time, an optimal solution should reduce the times worker agents return to the fuel pump.

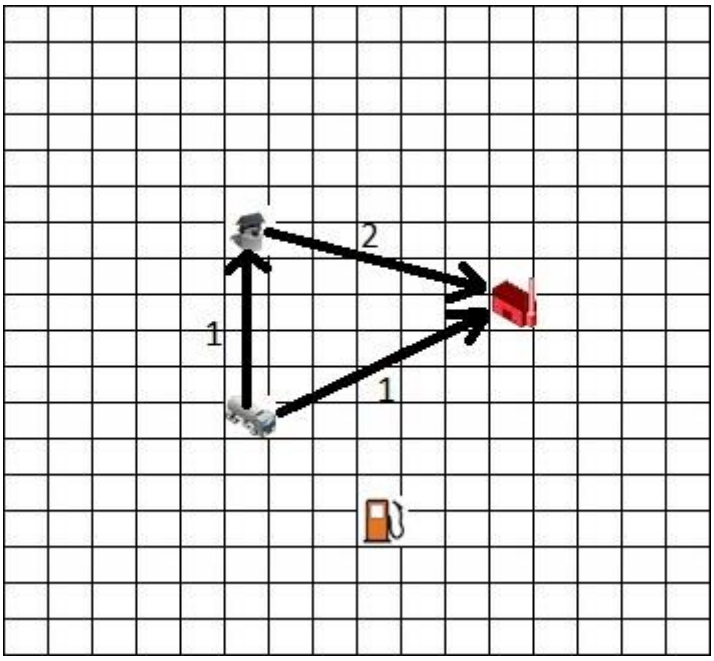

Figure 10 two optimal paths for finishing a task

There are many ways to finish a task. However, among them there are **two optimal paths** for finishing a task (illustrated by figure 10). These two paths can guarantee that the agent can finish a task without going back. Therefore, if there is such solution, the manager will allocate the solution to the agent immediately.

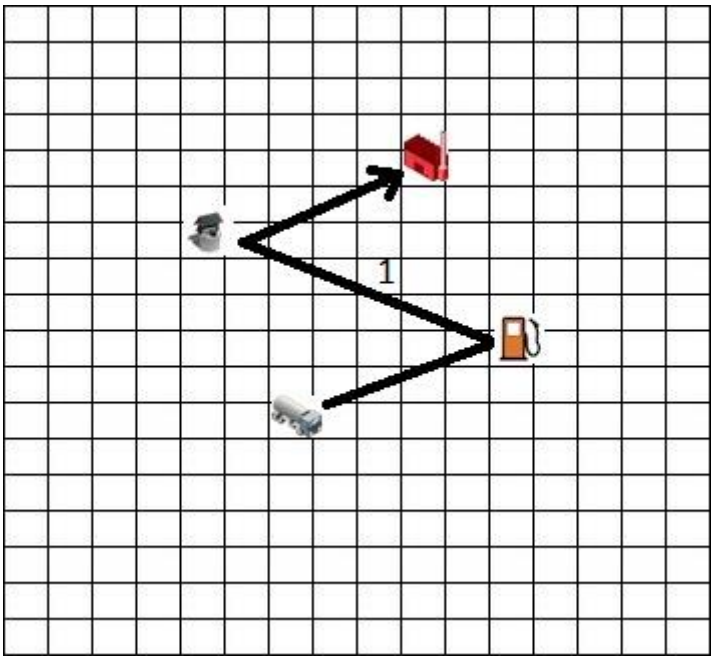

Figure 11 a bad path for finishing a task

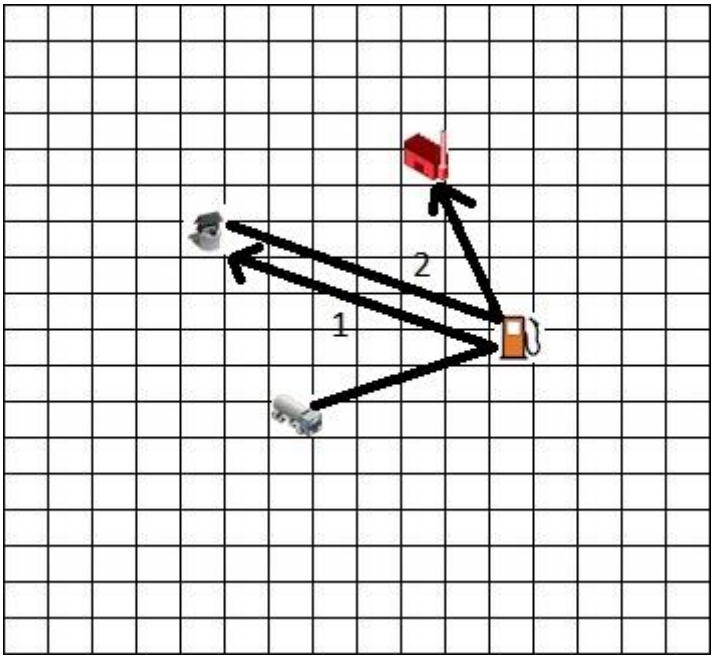

Figure 12 a worst path for finishing a task

Figure 11 and 12 illustrate two bad cases for finishing a task. Both of two solutions have to return to the fuel pump at least once. Therefore, if there is not any better solution, a solution with the shortest path (**lowest cost**) will be chosen.

Stations	Wells
(20,30)	(10,15)
(32,43)	(23,41)
(12,65)	(74,34)
.	.
.	.
.	.
.	.

Figure 13 the closest well for stations

As mentioned in the previous coursework, in order to decide a well for loading water, each station will be paired with the nearest well in advance (see figure 13). Therefore, for a given task, the place for loading water is already fixed.

4. Implementation

Overall Class Diagram

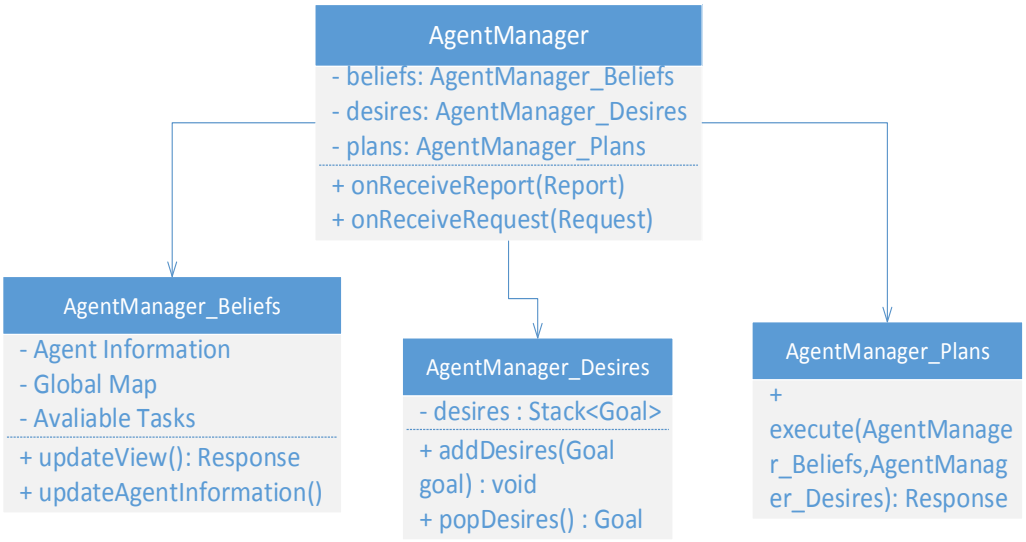

Figure 14 relevant class diagrams for AgentManager

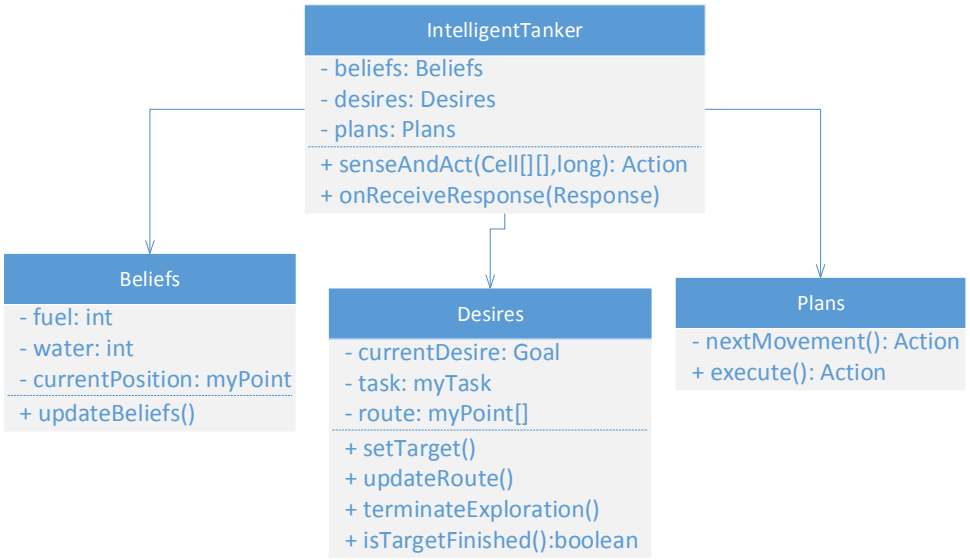

Figure 15 relevant class diagrams for worker agents

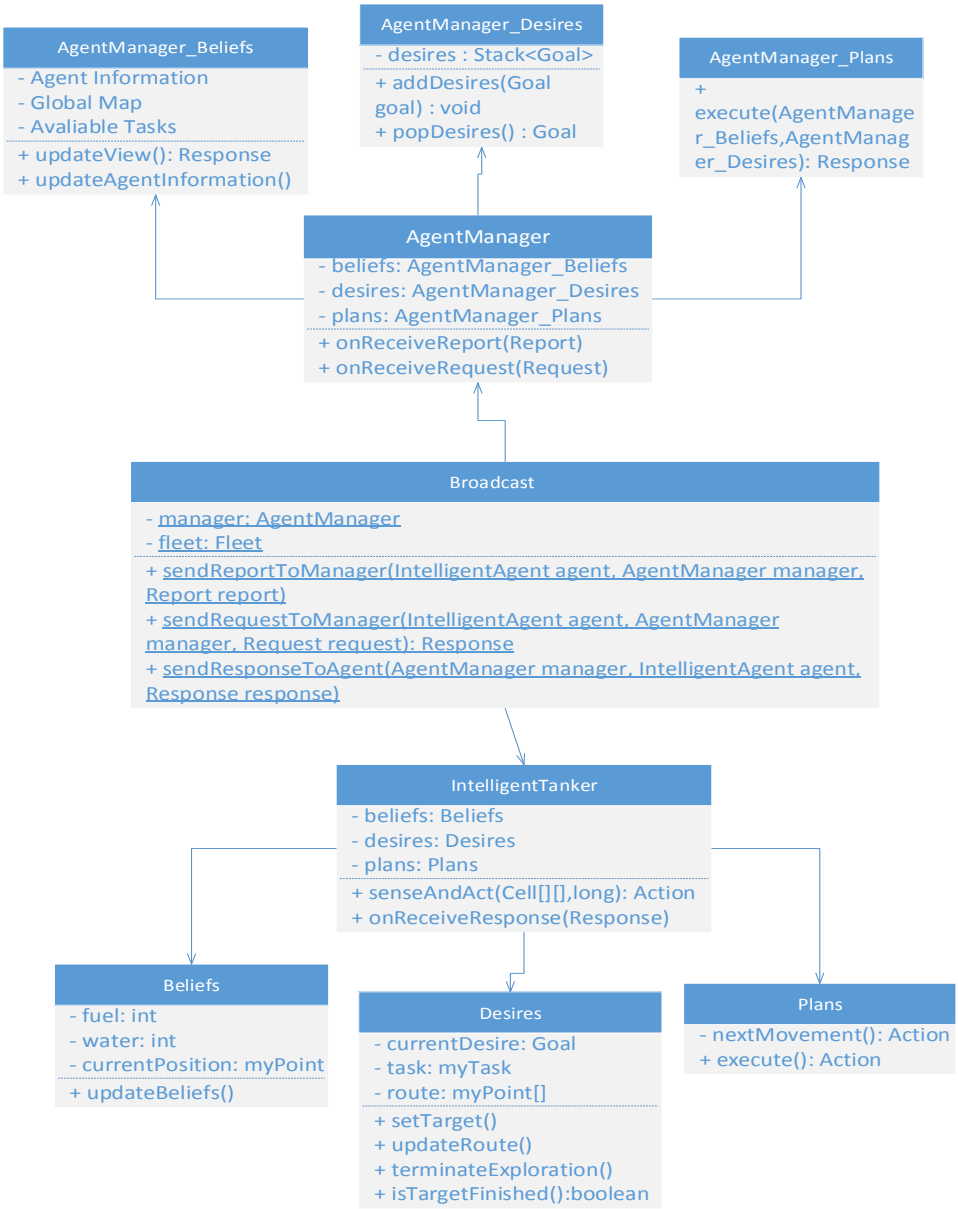

Figure 16 system overall class diagram

As many classes are very complex in detail, only key functions and key variables are displayed in diagrams above. Besides, some fundamental data structure classes (e.g. myPoint, myTask, Goal) are also left out for clarity purpose.

Instance Creation

Before we are able to run the program, we have to create instances for "AgentManager" and "Broadcast" respectively in the simulator.

```
AgentManager manager = new AgentManager((Tanker.MAX_FUEL/2)-5);

Broadcast broadcast = new Broadcast(manager, fleet);
```

Communication

Designing a method enabling communication between worker agents and manager can be quite easy. For example, the following demonstrates one possible way:

```
AgentManager manager = new AgentManager((Tanker.MAX_FUEL/2)-5);
IntelligentTanker tank = new IntelligentTanker(manager);
```

In the example above, worker agent class simply contains manager class as an internal field. Therefore, passing messages to manager would be simply calling functions in the manager class (e.g. manager.getMessage()).

However, there are two shortcomings of this method. First of all, this method makes the code not elegant. In addition, this method will easily confuse readers with that the relationship between worker agents and the agent manager is a "composition" relationship instead of "control".

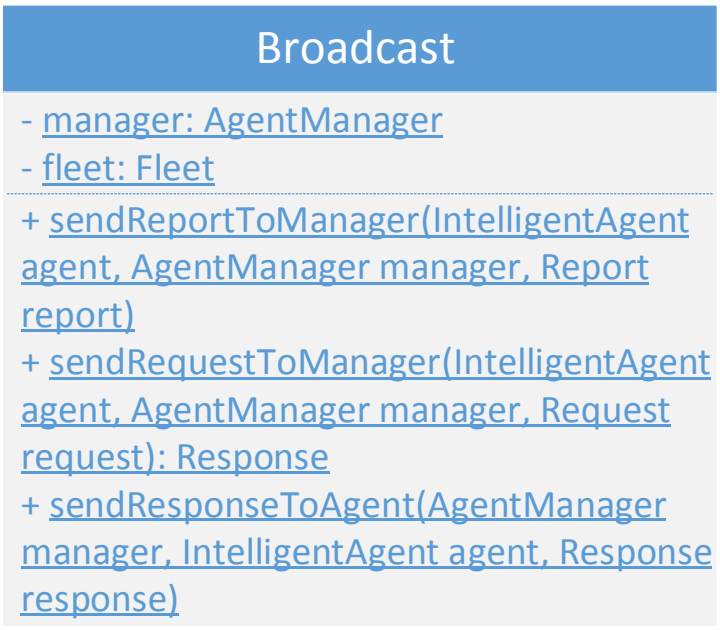

Figure 17 Broadcast class

In order to solve this problem, a "Broadcast" class is used to do such message passing (see figure 17). As described in "instance creation" section, "Broadcast" class constructor takes "AgentManager" and "Fleet" instances as parameters. In addition, all functions in "Broadcast" class are static functions. This simplifies the usage of functions in "Broadcast" class by simply calling "Broadcast.sendReportToManager(xxx)" without needing to get an instance.

Moreover, inspired by Android's "Intent", relevant "context" is required by relevant functions as something like "identity". For example, "sendReportToManager()" and "sendRequestToManager()" can never be used by the agent manager as it can never get an instance of worker agents and vice versa.

5. Evaluation

How is Performance?

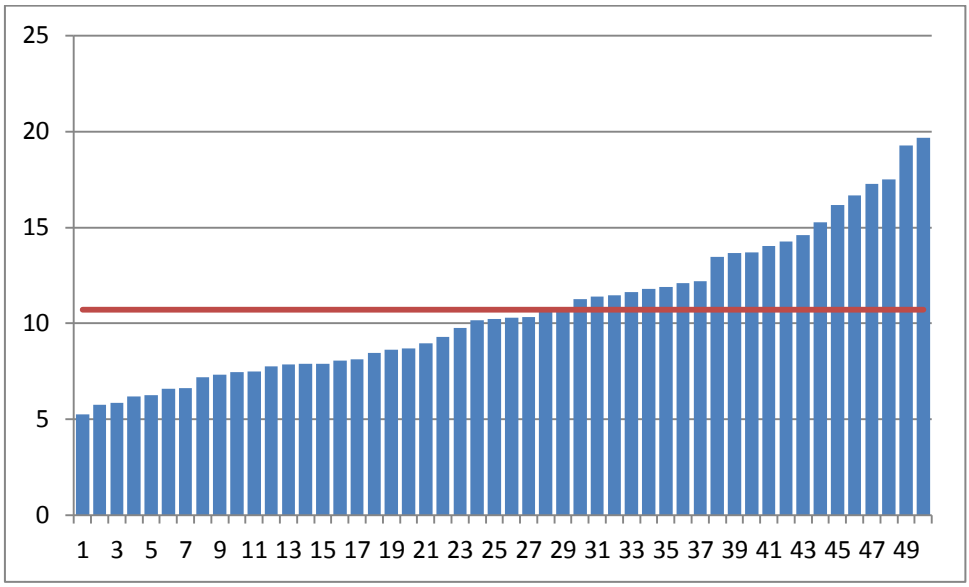

Figure 18 average scores of 50 experiments (unit: billion)

Figure 18 illustrates the score distribution of 50 experiments and the average score of these 50 experiments is about **10 billion per agent** (2 agents in total). Detailed data is attached in the appendix.

Factors Related to System's Performance

Generally speaking, there are three factors that can affect the performance of the whole system, the number of worker agents, how exploration points are allocated and how tasks are allocated. The following three sections will introduce corresponding experiments to each factor.

1. Quantity of worker agents

   As the total amount of stations and tasks is fixed if given a generated environment, it suggests that the more agents we use for this system, the higher total score the system will achieve. However, as mentioned in the specification, the performance of the system is measured by the average score shared by all agents. Therefore, the average score can be low although the total score are pretty high. In order to clearly present such "total score – average score" relationship and also eliminate the impact of different environment, we need to use some data from the environment and therefore need to make some modifications to the task environment.

```
/**
 * List of stations
 */
public ArrayList<Station> stations;

public int totalTasks = 0;

protected void generateTask() {
    if (Math.random()<NEW_TASK_PROBABILITY) {
        this.totalTasks++;
        if (this.task==null) {
            this.task=new Task(this);
        }
    }
}
```

Figure 19 modifications to the task environment file

Figure 19 shows the modification. By doing this, we can calculate the total amount of tasks in theory, and then get the "completion rate" (i.e. completed/total). "Completion rate" can be a much more stable parameter for measuring the performance of the system as it eliminates the impact of the environment.

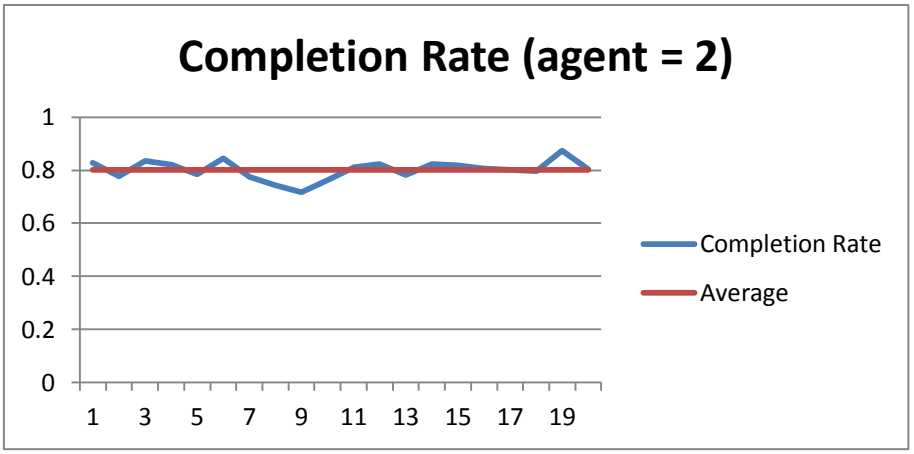

Figure 20 completion rates when there are 2 agents

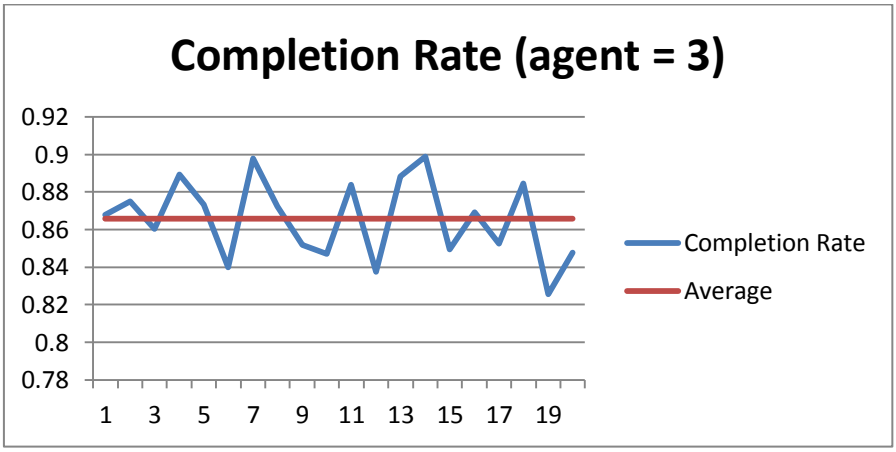

Figure 21 completion rates when there are 3 agents

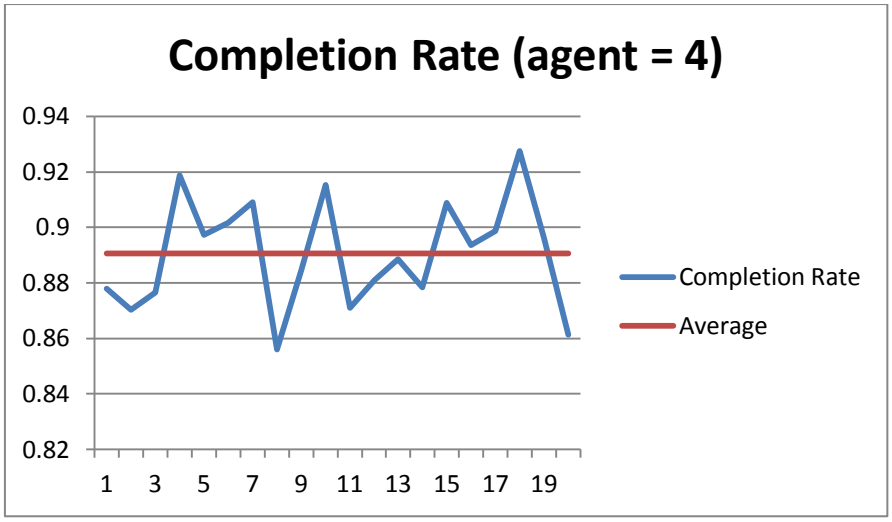

Figure 22 completion rates when there are 4 agents

Three figures above illustrate the average completion rate of the system in 20 experiments with different number of agents. Detailed data is attached in the appendix.

We can see that with more agents used in the system, the system will be able to complete more tasks. When there are 4 agents, the system can finish nearly 90% of total possible tasks. However, the average score per agent goes in opposite way. The system with only 2 agents can achieve almost 10 billion average score, while that with 4 agents can only achieve 6 billion average score.

Therefore, in order to get higher performance, only **2 agents** are used in this multi-agent system.

2. Exploration point allocation

As mentioned in the "Design" section, there are two ways of allocating exploration points: by distance or by station density. However, the experiment shows that these two methods seem to make **no difference** in terms of both completion rate and average score. Detailed data is attached in the appendix.

As selecting out the closest point needs a little bit more computation than the latter method, we will adopt the latter method for this system.

3. Task allocation

Replacing current task allocation module with a simpler version (allocating tasks in original order), we found it approximately decreases the average score of the system by 1 to 2 billion (see data in the appendix). However, based on the observation from

several repeated experiments, this kind of difference is not stable, especially compared to the difference caused by the agent quantity.

Based on previous three experiments, we can conclude that the performance of this multi-agent system is mainly affected by the number of agents used by the system.

6. Conclusion

This report demonstrates a hierarchical structure of multi-agent system solution to the water delivery problem under given task environment. Compared to the single agent system used to solve the same problem, multi-agent system achieves higher total score but lower average score.

Some factors like the number of worker agents and how exploration points and tasks are allocated may affect the overall performance of this system. Among them, the agent number has the greatest impact on the performance.

Finally, as this multi-agent system is designed based on previous single agent system which is limited to environment with some critical features (deterministic, partial dynamic and discrete), same restriction applies to this multi-agent system as well.

7. Appendix

Agent Number	Total Tasks	Completed	Completion Rate	Average Score
2	2162	1832	0.847363552	8468847772
2	3040	2371	0.779934211	14252115379
2	3248	2551	0.785406404	16184332259
2	2986	2322	0.777628935	13689313848
2	1972	1548	0.784989858	5867519850
2	2237	1800	0.804649084	7890665400
2	2645	2044	0.772778828	10289083112
2	2542	2145	0.843823761	11393931120
2	2512	2028	0.807324841	10233762552
2	2260	1870	0.827433628	8637574880
2	3246	2640	0.813308688	17510704200
2	2216	1874	0.84566787	8969574924
2	1941	1594	0.821226172	6273724178
2	2432	2004	0.824013158	10156550556
2	1778	1477	0.830708661	5270599911
2	1872	1622	0.866452991	6576863703
2	2838	2216	0.780831572	12181394104
2	2804	2343	0.835592011	13666826778
2	2622	2023	0.771548436	11628694250
2	2382	2023	0.849286314	10326133429
2	2346	1929	0.822250639	9285209671
2	2667	2054	0.770153731	10815388350
2	3080	2423	0.786688312	14609742607
2	2260	1785	0.789823009	7893265537
2	2658	2101	0.790443943	11279751103
2	3180	2467	0.775786164	15256777881
2	2665	2182	0.818761726	11804801106
2	2195	1717	0.782232346	7341663639
2	3328	2616	0.786057692	17281990548
2	2357	1792	0.760288502	8076685568
2	3419	2592	0.758116408	16676775072
2	1743	1515	0.86919105	5771842455
2	2955	2206	0.746531303	11889623050
2	2125	1758	0.827294118	7858191438
2	2510	2069	0.824302789	10602181872
2	2959	2321	0.784386617	13479684465
2	3701	2805	0.757903269	19670225190
2	1953	1714	0.877624168	7471505113
2	3110	2374	0.763344051	14018069981
2	2743	2221	0.809697412	12092076809
2	2561	2130	0.831706365	11477821305

2	3692	2766	0.749187432	19268523030
2	2071	1644	0.793819411	6610187802
2	2529	1991	0.787267695	9745772778
2	1816	1555	0.856277533	6205468525
2	1956	1689	0.863496933	7193905341
2	2343	1748	0.74605207	7499249498
2	2138	1764	0.825070159	7754401116
2	2347	1810	0.771197273	8112982910
2	2253	1866	0.828229028	8689635450
Average	2547.9	2038.62	0.804463002	10704032228

Table 1 result of 50 experiments

Agent Number	Total Tasks	Completed	Completion Rate	Average Score
4	2319	2036	0.87796464	5114381100
4	2576	2242	0.870341615	6196636896
4	3353	2939	0.876528482	10995793851
4	1697	1559	0.918680024	3019749871
4	2455	2203	0.897352342	6119149732
4	1891	1705	0.901639344	3722779692
4	1672	1520	0.909090909	2913690660
4	2370	2029	0.856118143	5106650099
4	2654	2348	0.884702336	6794407600
4	2422	2217	0.915359207	6028609396
4	3007	2619	0.870967742	8410453627
4	2702	2380	0.880829016	7102563790
4	3067	2725	0.888490381	9520933362
4	2789	2450	0.878451058	7529493125
4	2215	2013	0.908803612	5010533137
4	2471	2208	0.893565358	6119894832
4	2035	1829	0.898771499	4134950159
4	2389	2216	0.927584763	6240323034
4	2483	2225	0.896093435	6014058743
4	2509	2161	0.861299322	5765987223
Average	2453.8	2181.2	0.890631661	6093051996

Table 2 result of 20 experiments with 4 agents

Agent Number	Total Tasks	Completed	Completion Rate	Average Score
3	2711	2353	0.867945408	9159141129
3	2533	2216	0.874851954	8117284082
3	3136	2698	0.860331633	12057618310
3	2255	2005	0.889135255	6706471033
3	2376	2075	0.873316498	7271902516
3	2188	1838	0.840036563	5636094664
3	1968	1767	0.897865854	5080984940
3	2580	2250	0.872093023	8317802250

3	2692	2293	0.851783061	8613830296
3	3325	2816	0.846917293	12986341632
3	2804	2478	0.883737518	10319173396
3	2858	2394	0.837648705	9527924490
3	1860	1652	0.888172043	4675903950
3	2075	1865	0.898795181	5855569096
3	2227	1892	0.849573417	6010073593
3	1872	1627	0.869123932	4491501623
3	3643	3106	0.852594016	16196972086
3	1759	1556	0.884593519	4023315486
3	2886	2383	0.825710326	9367867697
3	3476	2947	0.847813579	14230444130
Average	2561.2	2210.55	0.865601939	8432310820

Table 3 result of 20 experiments with 3 agents

Agent Number	Total Tasks	Completed	Completion Rate	Average Score
2	1499	1240	0.827218145	3903699180
2	2043	1588	0.777288302	6309790960
2	2557	2139	0.83652718	11285421753
2	1898	1560	0.821917808	6023121780
2	2853	2238	0.784437434	12396886260
2	2377	2006	0.843920909	10277961660
2	2346	1820	0.775788576	8123257870
2	3370	2503	0.74272997	15668373262
2	2759	1977	0.716563972	9897112090
2	3337	2546	0.762960743	15977631772
2	1893	1537	0.811938722	5963818216
2	2517	2073	0.823599523	10768075156
2	2926	2292	0.783321941	12867462192
2	1521	1254	0.824457594	4029634323
2	2330	1907	0.818454936	9087269772
2	2818	2271	0.805890703	12641671386
2	3097	2486	0.802712302	15327657983
2	2786	2221	0.797200287	12315340613
2	1645	1438	0.874164134	5145951305
2	2554	2054	0.804228661	10529113127
Average	2456.3	1957.5	0.801766092	9926962533

Table 4 result of 20 experiments with 2 agents

Agent Number	Total Tasks	Completed	Completion Rate	Average Score
2	2633	2138	0.812001519	11461064355
2	2853	2358	0.826498423	13849749549
2	1825	1568	0.859178082	6211199232
2	2457	1913	0.778591779	9126118583
2	2083	1648	0.791166587	6723523584
2	2013	1690	0.839542971	7046551330

2	2327	1876	0.806188225	8926089606
2	1656	1385	0.836352657	4773817307
2	3111	2263	0.727418836	12931972338
2	3277	2558	0.780592005	16254464391
2	2772	2216	0.799422799	12244301912
2	2393	1902	0.79481822	9016164720
2	2186	1762	0.806038426	7788839067
2	2443	2089	0.855096193	10813216540
2	2867	2142	0.747122428	11491039602
2	2257	1761	0.780239256	7875583822
2	2704	2208	0.816568047	12381377664
2	2277	1882	0.826526131	8948378335
2	2680	2035	0.759328358	10282654552
2	2833	2270	0.801270738	12771409305
Average	2482.35	1983.2	0.802198084	10045875790

Table 5 result of 20 experiments (2 agents, allocating exploration point by distance)

Agent Number	Total Tasks	Completed	Completion Rate	Average Score
2	2183	1787	0.818598259	8125115517
2	2028	1631	0.804240631	6461238304
2	1955	1572	0.804092072	6234998448
2	2326	1792	0.770421324	8084359808
2	3065	2210	0.721044046	12226862570
2	2227	1682	0.755276156	7055275150
2	1429	1219	0.853044087	3761067858
2	2275	1771	0.778461538	7800809593
2	1392	1173	0.842672414	3373169707
2	2475	1975	0.797979798	9803895062
2	2487	1989	0.799758745	9827754417
2	2762	2152	0.779145547	11624868356
2	2273	1839	0.809062912	8475624577
2	2864	2381	0.831354749	14219411763
2	2374	1937	0.815922494	9477492095
2	2062	1680	0.814742968	7067106480
2	2115	1691	0.799527187	7068120431
2	2755	2230	0.809437387	12402562010
2	2507	1965	0.783805345	9594529080
2	1906	1574	0.825813221	6098093897
Average	2273	1812.5	0.800720044	8439117756

Table 6 result of 20 experiments (2 agents, allocating tasks in original order)

8. References

1. BERGENTI, F., GLEIZES, M.-P., & ZAMBONELLI, F. (2004). *Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook*. Boston, [Mass.], Kluwer Academic.

2. LIANG, Y. D. (2001). *Introduction to Java programming*. Upper Saddle River, N.J., Prentice Hall.

3. RUSSELL, S. J., & NORVIG, P. (1995). *Artificial intelligence: a modern approach*. Englewood Cliffs, N.J., Prentice Hall.

4. VLASSIS, N. (2007). *A concise introduction to multiagent systems and distributed artificial intelligence*. [San Rafael, Calif.]: Morgan & Claypool Publishers.

5. WEISS, G. (1999). *Multiagent systems*. Cambridge, Mass.: MIT Press.

6. WOOLDRIDGE, M. J. (2002). *An introduction to multiagent systems*. New York, J. Wiley.