Final Report

Josephine Krause[1], Marc Schmidt[1] and Muzammal Hussain[1]

[1] Technische Universität Berlin, 10623 Berlin, Germany

Abstract. The following paper is the final report for the Application System Project B in the summer term 2018. The project concentrates on finding a decentralized solution for the Multi-Agent Programming Contest 2018. In order to implement a solution, a strategy had to be defined. In this approach, a team-oriented strategy was chosen. The coordination of the agents of the simulation was done by using an auctioning process similar to the Contract Net Protocol. For the evaluation of the approach, different metrics concerning the performance and the success were measured and processed.

Keywords: Multi-Agents, Decentralized Systems, Auctioning

1 Introduction

Working well as a team requires coordination and strategy as well as a common goal. This does not only apply to people who are trying to solve a problem together but it also applies to systems that are using multiple agents. If one is looking for a decentralized approach in such a system, one has to ensure that the agents are able to coordinate in line of their strategy.

An example for such a system is the simulation of the Multi-Agent Programming Contest 2018 [1]. The scenario given in this contest states that a team of agents must work together in order to win the simulation. There are four different kinds of agents, drones, trucks, cars and motorcycles, and they have different properties. Furthermore the simulation is providing different facilities like stores, shops, dumps, charging stations and resource nodes. Additionally, it provides different kinds of items. In order to find the items and their resource nodes, the agents firstly need to explore the city.

The items are needed for different jobs that can be done by the agent. A job consists of delivering a certain amount of items to given stores. The server has been configured so that the assembly of items will not be requested. When a job is done, the team will receive a reward in the form of Massiums, the currency of the simulation. The team can use this money to upgrade their skills or to build wells.

There are different kinds of jobs that can be done: mission jobs, priced jobs and auction jobs. Mission jobs are the only jobs a team has to do. The team is assigned to this mission job. If it fails to do the job it has to pay a fine. Priced jobs can be done by any team. The first team that finishes the job receives the payment. The third type of job, auction jobs, can be done exclusively by one team if it wins the auction concerning this job.

Wells are generating points because it is the goal of the simulation to solve a water crisis in the year 2045 A.D. In a contest between two teams the one that earns the most points wins.

In order to implement an elaborate approach, a strategy for the team had to be found. This was already done once in the milestone report. Due to some reconsideration, the final strategy is differing from the initial strategy. Hence, the final strategy will be explained in the second chapter.

In the third chapter the final implementation approach will be described. An evaluation of the implementation will be done in chapter 4.

Finally, the structure of the team will be briefly explained in the fifth chapter.

2 Final Team Strategy

The final strategy of the team is differing slightly from the strategy that was planned beforehand in the milestone. This includes the overall strategy. Initially, a specialization strategy was followed. But due to the need of faster accomplishments instead a team-oriented strategy was chosen. Instead of defining roles and dedicating the different kinds of agents to different kinds of assignments, e.g. using drones to explore the city and trucks to hoard the items, now all agents are doing all assignments.

Subsequently, there was a shift in the different aspects of the overall strategy. Those different aspects can be categorized in the following manner: Exploration, Defining Roles, Sharing Information, Massium, Items, Jobs, Auctioning, Charging, Idle Time, Upgrades, Wells and Failures.

The Exploration is done by using a grid. The agents choose the closest, unexplored waypoint. Due to the team-oriented strategy, the exploration is done by all agents. This ensures a faster exploration.

As stated before, defining the roles of the agents has been reconsidered. Since now a team-oriented strategy is pursued, there are no roles to be defined. All tasks can be done by all agents.

Sharing information is crucial to the functionality of the team. Therefore, information about the items, the facilities, the wells, the agents and the visited grid points are shared among the agents.

Massium, the currency of the game, is needed in order to build the wells and to upgrade agents. Originally, it was planned to save the earned money. It was supposed to be spent in a considerate manner. This strategy has changed since the simulation has a limited amount of steps. When saving money up to a certain point, there is the risk that the agents do not have enough time to build up the wells and let the wells generate enough points. Hence the money is spent on demand. As soon as there is enough money to build a well, the building of the well is triggered.

Initially, it was planned to let the trucks gather items. Following the new strategy, the gathering of the items is done on demand by all the agents. If an item is needed for a job, it is gathered.

The handling of the jobs is mainly following the original strategy since it does not necessarily require a specialization strategy. All the agents are committing to the jobs.

The jobs on the other hand are decomposed into tasks so they can be done by multiple agents. Initially, a cost-benefit analysis of the jobs was planned but in order to improve the efficiency, all jobs, except for the auction job which is not handled yet, are done. If a job cannot be finished in time, the items required for the job are still stored in the corresponding store, so the agent can empty its load.

Auctions are used for the coordination. Every job is assigned to an auctioneer. The auction is done in stages. The job is decomposed into different tasks divided by the items. It is possible that the tasks are linked since they can be depending on a foregoing task. For example, if one task is the gathering of an item and another task is the delivery of that item to a store, they should be done by the same agent.

Since the battery of an agent is discharging during the actions, the charging must be considered in the strategy. The charging is triggered in every state except for the plan execution state because the plan execution should not be interrupted and therefore delayed.

If an agent is not executing a plan it is having some idle time. If that is the case, the agents have the following priorities: firstly, they explore the city and then they start building wells.

The initial strategy of specialization also considered that the skills of the agents are upgraded depending on the role. Since this strategy was reconsidered, updates are currently not done. This also aids the saving of the money. The risk that too much money is spent on updates instead of on the wells must be avoided.

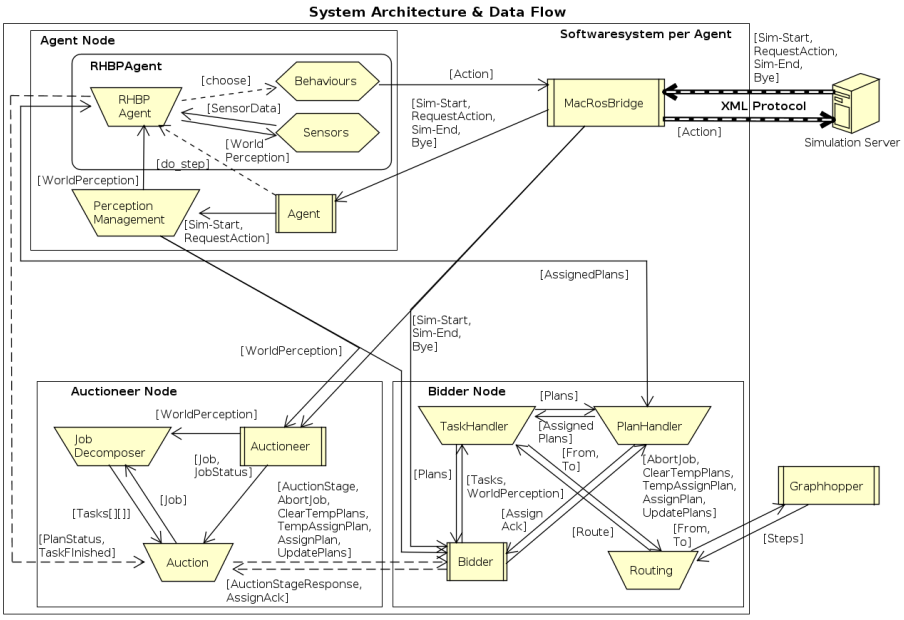
Initially it was planned to build the wells on the borders of the map. In order to avoid predictability this strategy has been discarded. Now the wells are placed only in positions where trucks have been before. This prevents that an agent might not reach a chosen position. Furthermore, only one well at a time is built. Hence, conflicts in the management of the Massiums are avoided. Wells are only built if enough money is available. If two agents started to build wells at the same time, one of them could end up not having enough money. Additionally, it is ensured that a well is build up completely so that it generates points before the building of a new well is started.

The failure of actions is handled mainly considering the exploration. If a grid point cannot be reached by some agents, because the roads might not lead to it, the action fails. If that is the case, a new grid point is assigned. The old grid point will be considered as explored and not be visited again.

To sum it up, the final strategy focuses on the cooperation of the agents without considering specific roles.

3 Description of the Final Implementation Approach

To initiate the description of the final implementation, the architecture of the system as it can be seen in the Graphic 1 must be explained. Generally, the system consists of the following components: Simulation Server, MacRosBridge, Agent Node, Auctioneer Node, Bidder Node and the Graphhopper. Graphhopper is tool used to calculate the rout planning [2].



Graphic 1 System architecture and data flow

The Simulation Server is providing the information of the simulation to the MacRosBridge using the XML Protocol.

The MacRosBridge can forward the information of the simulation to the Agent Node. The Agent, which also contains a RHBPAgent, has an Agent that receives the information from the MacRosBridge. In turn the Agent sends the information it gained from the MacRosBridge to the Perception Management. Here the world perception is managed.

The perception of the world is send to the RHBPAgent, the Auctioneer and the Bidder, so all of them are sure to have the current world perception. This avoids failures which are due to incorrect information.

The RHBP Agent is component for ROS used for planning and decision making. It is formed by a reactive behavior network and a symbolic planner [3].

The world perception, which is sent to the Auctioneer Node and the Bidder Node by the MacRosBridge, also contains information concerning new jobs.

The MacRosBridge also can inform the Auctioneer Node about an incoming job. The coordination of an incoming job is done with an auction. Every agent can be an auctioneer. By using the IDs of both the agent and the job as well as the overall amount of agents, the auctioneer is determined. This prevents that one agent might always be the auctioneer.

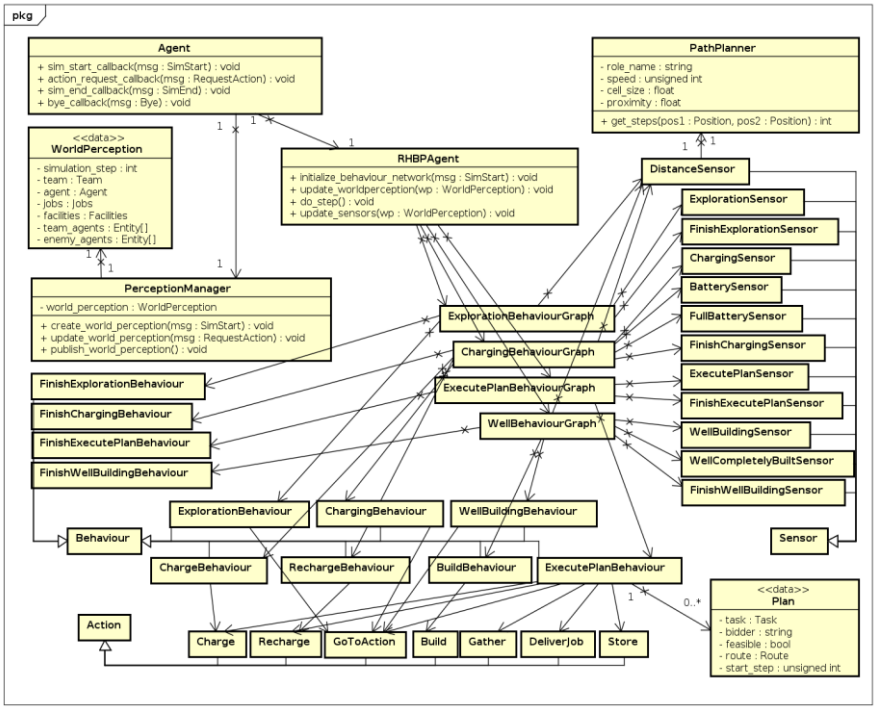
The components Auctioneer Node and Bidder Node are needed for the coordination of the agents. As stated before the auctioneer receives information about an incoming job from the world perception it got from the MacRosBridge. The job is then split into tasks in the Job Decomposer. The tasks are then spread in the auction.

When a Bidder Node receives the task from an Auction, it elaborates a plan using its TaskHandler which in turn needs the Routing and the Graphhopper because the way an agent has to cover is part of the plan.

The plan is used to bid for the auction. The Auctioneer chooses the bidder whose plan has the lowest end step. Sometimes the duration of another plan might be shorter. However, it must be considered, that the agent with the shortest duration could be busy. Hence the agent might finish later even though the duration of the task is shorter. Therefore the lowest end step was chosen.

Once the bidder was chosen and is assigned to a task, the assignment must be acknowledged by the bidder. When this is done, the assignment is given to the PlanHandler. The assigned plans are given to RHBPAgent in the Agent Node. The RHBP chooses a behavior. The behavior results in an action which is given to the MacRosBridge. In turn, the MacRosBridge forwards the action to the Simulation Server so it can be processed.

The class diagrams in the Graphics 2, 3 and 4 display the components of the different nodes used in the system.



Graphic 2 Class Diagram of the Agent

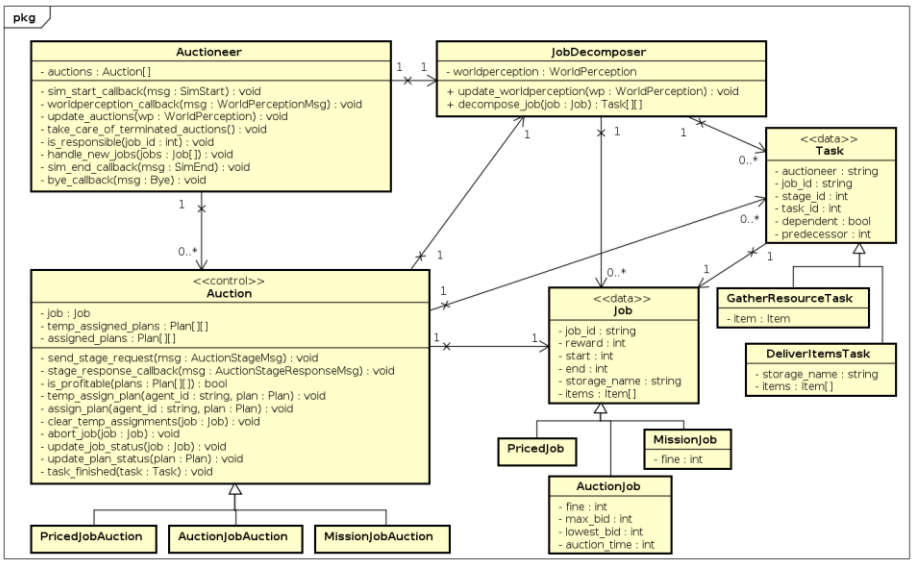
The Class Diagram of the Agent is shown in the Graphic 2. Every Agent has a RHBPAgent and a PerceptionManager which in turn has a world perception.

The RHBPAgent is using Graphs for the exploration behavior, the charging behavior, the execute plan behavior and for the well behavior. The Graphs are using differ-

ent sensors like the DistanceSensor and the BatterySensor. More sensors are those that indicate a running behavior or the finishing of a behavior like the Exploration-Sensor and the FinishExplorationSensor.

The classes ExplorationBehaviorGraph, ChargingBehaviorGraph, ExecutePlanBehaviorGraph and WellBehaviorGraph have the corresponding ExplorationBehavior, ChargingBehavior, ExecutePlanBehavior and the WellBehavior as well as the FinishExplorationBehavior, FinishChargingBehavior, FinishExecutePlanBehavior and FinishWellBuildingBehavior. The behavior graphs are used to declare all the sensors and behaviors. They also have more behaviors, for example the ChargeBehavior. Each behavior graph has a predefined goal. In order to structure the various functionalities the usage of the behavior graphs were chosen instead of a single class. Once the finishing behaviors are triggered, the agent goes into the idle state.

The ExecutePlanBehavior must have a Plan it can execute. The Plan has the Actions which can be Store, Build, Gather, DeliverJob, GoToAction and Charge. Those actions are needed for the execution of a plan.



Graphic 3 Class Diagram of the Auctioneer

The class diagram of the Auctioneer is displayed in the Graphic 3. The Auctioneer uses a JobDecomposer to split the jobs into tasks. The tasks can be depending on one another as stated before. A task can be a GatherResourceTask or a DeliverItemTask. The former is about gathering an item needed for the job, the latter is about delivering said items to a store. Since those tasks are link, an agent only can take part in a job if it first gathers the amount of the item type demanded by the job. Then it delivers that gathered item to corresponding storage. Hence, the two tasks are being done in row, depending on each other. Each task belongs to a job. Jobs can be PricedJobs, Mission-

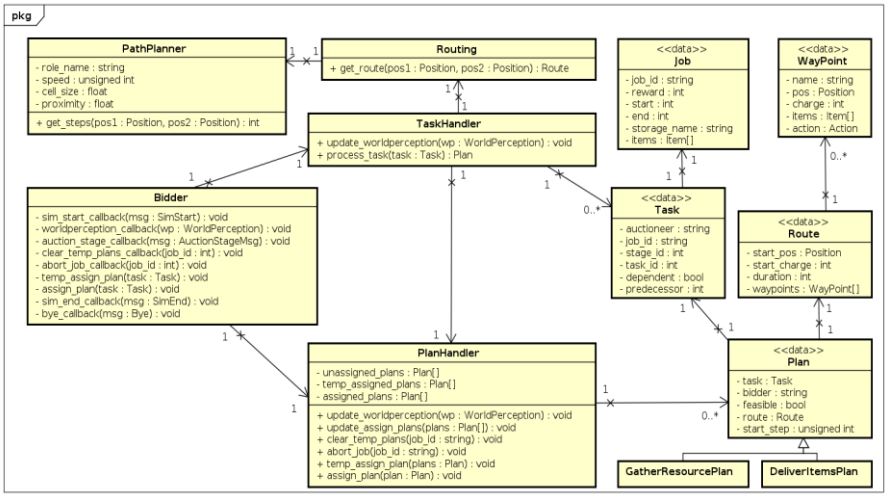
Jobs or AuctionJobs. The jobs are split into tasks in order to allow multiple agents to do parts of the job. This accelerates the execution of the jobs.

The Auctioneer has several Auctions which in turn has the Job the auction is for, the JobDecomposer and the Tasks provided by the JobDecomposer. An Auction can be a PricedJobAuction, an AuctionJobAuction or a MissionJobAuction, depending on the job that needs to be coordinated through the auction.

In Graphic 4 the class diagram of the bidder is shown. Each bidder has an instance TaskHandler as well as a PlanHandler.

The TaskHandler has a class dedicated to the Routing which is using the Graphhopper. Additionally, it creates Tasks for a certain job. The TaskHandler also has the PlanHandler.

The PlanHandler has three different lists of Plans: unassigned plans, temporary plans and assigned plans. Only the assigned plans will be sent to the RHBP Agent and be executed there. Plans can be either a GatherResourcePlan or a DeliverItemPlan. Those plans contain a Route which has various WayPoints.



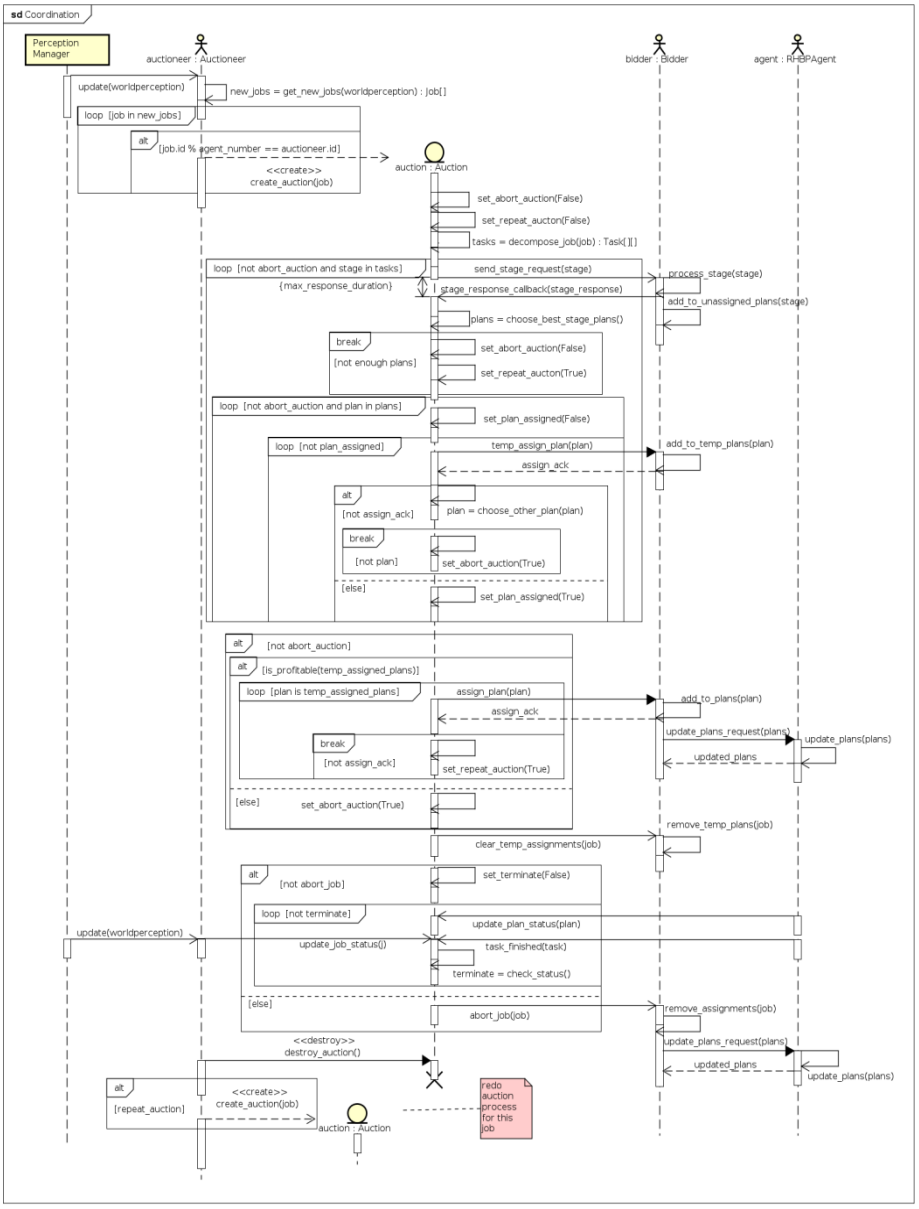
Graphic 4 Class Diagram of the Bidder

The coordination of the agents is shown in the sequence diagram in Graphic 5. Auctions equating the contract net protocol (CNP) are used for the coordination of the agents. The CNP was developed by Reid G. Smith [4]. It is a high-level protocol for nodes of distributed systems and enables cooperative task execution. This is a good approach to conquer the MAPC. The manager of the CNP sends out a task announcement to the other nodes. These announcements contain a time frame for the task as well as a task description. The bidders return a bid for the task announcement. If a bidder was successful, it is informed by the manager. For the time of the contract between the bidder and the manager, the bidder is called the contractor. As long as the task is executed, the contractor is giving current status information to the manager.

This CNP is similar to the auctioning process used in this approach. The PerceptionManager is updating the Auctioneers world perception. This can include infor-

mation about new jobs. The Auctioneer then creates an auction. Requests for the auction are sent to the bidders which process them and respond with the plans to the auctioneer.

If a bidder is chosen, it is temporarily assigned to the task. Only when each task is assigned temporarily and the assignments are acknowledges, a plan is assigned definite. If a definite assignment fails, the auction gets restarted.



Graphic 5 Coordination

4 Evaluation

In order to evaluate the proposed system, various metrics have been monitored. They have been grouped into the four different aspects Success, Performance, Jobs and Wells.

The aspect Success contains the metrics Score and Massiums left. Score is the final score reached in the simulation. Massiums left is the amount of money left at the end of the simulation.

The aspect Performance contains the metrics Mean Action Response Duration (MARD), Mean Decision Making Timeouts per agent (MDMT), Percentage Decision Making Timeout steps (PDMT) and Mean Auction Stage Response (MASR). Mean Action Response Duration is the average time in seconds that passes from sending a request to receiving the response of an action. Mean Decision Making Timeouts per agents is the average amount of timeouts that are happening while the agent is making its decision. A decision timeout occurs when the triggering of an action by a behavior is missing. Percentage Decision Making Timeout steps is the percentage of the overall steps that are timeouts from the decision making process.

The aspect Jobs contains the metrics Total number of jobs, Jobs done, Priced jobs done, Mission jobs done, Jobs failed, Priced jobs failed and Mission jobs failed. Total number of jobs describes all the jobs that are available in the simulation. Jobs done displays the sum of all the jobs that have been done by the team. It can be split into the metrics mission jobs done and priced jobs done, which describe how many priced or mission jobs have been done. Similarly, the metric Jobs failed describes the overall amount of failed jobs. This metric can be differentiated by the job types, too.

The aspect Wells contains the metrics Amount and Costs. Amount describes the amount of wells built by the team while Costs describes how much money in the currency Massium was spent to build these wells.

In order to gain reasoned results, the simulation was run with different random seeds and different amounts of agents. The random seed values were 18, 20 and 22, while the chosen amounts of agents was 8, 12 and 16. The amount of agents was chosen to be a multiple of four so that each type of agent could be used in equal amounts. Hence, three simulations were run with 8 agents, using the random seed values 18, 20 and 22. Three simulations were run with 12 agents, using the random seed values 18, 20 and 22. Consequently, three simulations were done with 16 agents, using the random seed values 18, 20 and 22.

The average metrics for the different amounts of agents over the course of the three simulations with the different random seeds 18, 20 and 22 are shown in Table 1.

In table 1 it is shown that the metrics of the aspect Performance (MARD, MDMT, PDMT, MASR) are in average increasing the more agents are being used. This indicates that the performance is getting worse when more agents are used. There are more timeouts and the responses are taking longer. The reason for this is that the more agents are used, the more processing needs to be done. Hence, the performance is decreasing.

Table 1 Mean Metrics by agents

Metric	8 agents	12 agents	16 agents
Score	54657	32866	94233,33
Massiums left	11314,33	5965,66	2621,33
MARD	0,85241	1,17444	1,67276
MDMT	6,6	5,6	36
PDMT	0,0066733	0,00567	0,036
MASR	0,071513	0,07923667	0,0973733
Total number of jobs	202	211	201,33
Jobs done	84,667	118,67	142
Jobs failed	0,333	1,33	3,667
Amount of wells	3,667	5,667	15
Costs of wells	5996,333	13235	23658,667

Diagram 1 displays the average metrics of the aspect success for the various amounts of agents over the different simulations. It shows that the more agents are used, the less money is left. Furthermore it shows, that the highest scores are aimed when using 16 agents. The low average score of the 12 agents is due to the simulation using the random seed value 20. In this simulation the team only earned 184 points. This value differs strongly from the scores of the remaining two simulations with 12 agents. When using the random seed value 18, the team scored 66030 points and when using the random seed value 22, the team earned 32384 points. Hence, the average of the score for the simulations using 12 agents is lower than the scores aimed in the other simulations using that amount of agents. But overall the trend shows that more agents are able to earn more points. This is due to the fact that more agents can do more jobs, thus earn more money and build more wells.

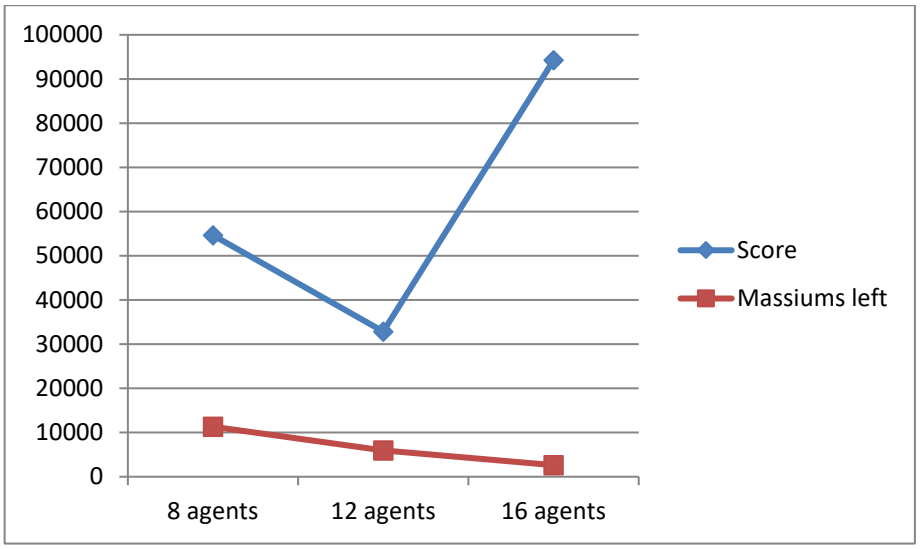


Diagram 1 Mean Success by agents

This is also shown in the amount of wells in Table 1. The more agents are being used, the more wells are being built. Consequently, the costs for the wells are also increasing.

The Diagram 2 shows what kind of jobs averagely have been done or failed more often. It can be seen that priced jobs are being done the most. They are failing rarely. Mission jobs are not done as much as priced jobs. This is due to the simulation. There are more priced jobs to be done than mission jobs. The Diagram 2 also shows that mission jobs failed very rarely. Since the teams have to pay a fine when they are not doing the mission jobs, they are having a higher priority. Since the server has been configured to not create jobs that require assembling, those jobs are not considered here.

Table 1 is showing how many jobs have been done or failed overall. Both metrics are increasing with the amount of agents. This is due to the fact that more agents are able to do more jobs. On average there were 200 jobs to be done in each simulation. Thus, not all jobs were done in the simulations.

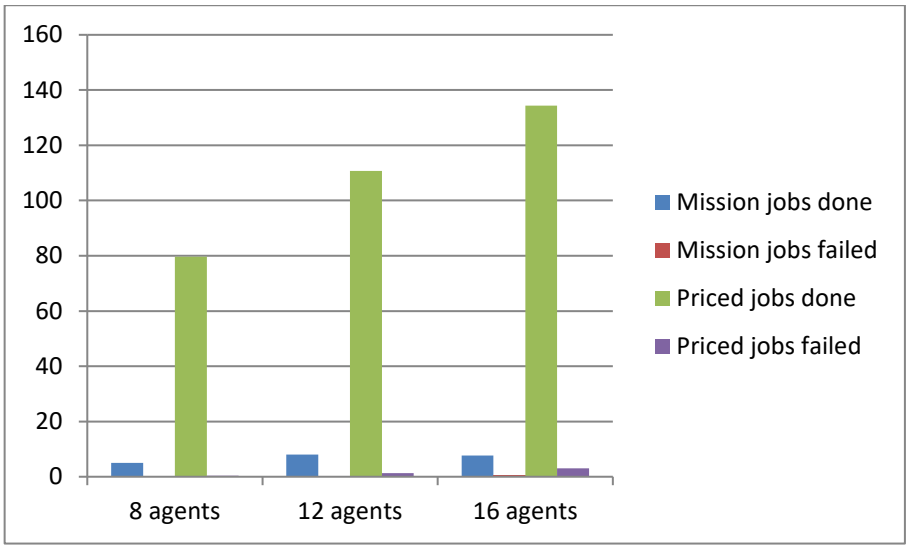


Diagram 2 Jobs by agents

To sum it up, the amount of agents influences the success of the simulation. Considering that in order to build wells and generate points, jobs have to be done to earn money, this is reasonable. Furthermore the coordination of the team, the fulfillment of the jobs is still working even if the team is growing up to 16 agents. Anyhow, the slight decreases of the performance must be considered too.
All in all it can be said, that the provided system is giving a working approach on a decentralized system using multiple agents.

5 Team Structure

The team consists of three members: Josephine Krause, Marc Schmidt and Muzammal Hussain. During this project their main responsibilities were the following:

Josephine Krause was the group coordinator. Her main responsibility was the monitoring of the progress as well as the preparation of the weekly progress presentation. Furthermore, she was responsible for the quality of the reports and the presentations.

Marc Schmidt was the head of programming. He was responsible for the decisions concerning the code, including the continuous improvements of the code. He also was providing help and support to the other group members when it came to the programming.

Muzammal Hussain was the scientist of the group. Doing the research necessary for this project was his responsibility, so all decisions could be made in a reasonable manner.

References

1. MASSim Scenario Documentation, https://github.com/agentcontest/massim/blob/master/docs/scenario.md, last accessed 2018/07/21.
2. Graphhopper Homepage, https://www.graphhopper.com/, last accessed 2018/07/21.
3. RHBP Homepage, https://gitlab.tubit.tu-berlin.de/hrabia/rhbp, last accessed 2018/07/21
4. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers C-29(12):1104-1113.