LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

BACHELORARBEIT

Action Trading for Self-Interested Multi-Agent Reinforcement Learning in a Smart Factory Setting

Arnold Unterauer

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR MOBILE UND VERTEILTE SYSTEME

BACHELORARBEIT

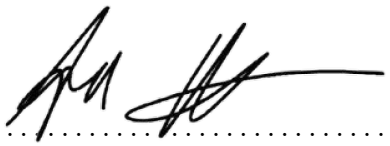
Action Trading for Self-Interested Multi-Agent Reinforcement Learning in a Smart Factory Setting

Arnold Unterauer

Aufgabensteller:	Prof. Dr. Claudia Linnhoff-Popien
Betreuer:	Kyrill Schmid Robert Müller
Abgabetermin:	18. März 2020

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 18. März 2020



(Unterschrift des Kandidaten)

Abstract

Cooperation is essential for all living beings, as it is a key component to success. Since multi agent systems have become more and more important in our daily lives, the demand for cooperating agents has also increased. Agents in multi agent environments try to maximize their reward as they interact with the environment, which mostly results rather in selfish than cooperative behaviour. Recent work on cooperation between entities in multi agent environments have introduced the concept of trading markets, which enables agents to interact with each other and thus enable cooperation. These trading markets enable agents to make an offer, consisting of an action that the other agent can follow. As the actions of the agents are extended with offers, they were able to outperform agents without this possibility. While cooperation between agents not only led to a higher overall reward, individual rewards also increased. These results have shown that the cooperation between agents with extended actions can have a positive impact on the reward and we therefore take a closer look at the cooperation emerging from the extended actions. Further investigating the behaviour of agents, we compare the short- and long-term cooperation with a modified version of action trading and a new trading mode. While the agents in the original action trading were only able to trade one action for reward, we want to scale this approach to n offer actions. We therefore split the trade into two separate parts, offer and supply. As we increase the amount of offer actions we will be able to examine the behaviour and reward of short- and long-term cooperation. In addition, we will analyze the cooperative behavior of agents as we change various factors that influence the trade, such as the payment timing, the amount of compensation or the trading budget. For this purpose we use reinforcement learning and deep q-networks to train the agents to maximize their rewards in the environment smart factory. In the smart factory the agents have to compete with each other in order to process machines for rewards. This competition encourage the agents to trade with each other and therefore create cooperation. Our results show that the adaptation of action trading also works in the environment smart factory. The new introduced trading mode reveals that agents rather cooperate short- than long-term, which is also reflected in the rewards. While the increase of compensation shows mixed results, the budgeting of the trade only has negative impact on the reward. Also the variation in the payment timing shows that agents are more likely to cooperate if they pay afterwards.

Contents

1 Introduction

Reinforcement Learning has shown huge success recently, as it is used in many branches of the industry. There the agent has to interact with an environment and make use of the resources in order to complete a goal, which is given to him. As he receives states of the environment, he learns what actions he has to execute to achieve the objective efficiently. If the single agent environment is extended to a multi agent one, where multiple agents have to complete goals in an single environment, new problems emerge. While the agent in single agent reinforcement learning only has to deal with the change of the environment caused by himself, in multi agent reinforcement learning all agents interact with the environment making a accurate prediction impossible. Additionally all of the resources are shared between all agents, which can create a bottleneck. As every agent wants to complete his goal in the fastest way possible, they want to utilize the resources as soon as they can. This selfish behaviour creates competition between agents, which results in unused potential. In order to prevent this competition between entities, it is desired to motivate the agents to cooperate. Recent work in the field of Multi Agent Reinforcement Learning have been focused on the cooperation between agents [BEH$^+$18], [Kra93], [LP17]. While this has originally been researched in the field of game theory [CEW11], many core elements have been adopted by the multi agent reinforcement learning community. In game theory real problems are broken down into mathematical models, which are easier to analyse. Furthermore these models, called games, can be divided into cooperative or non-cooperative games. In order to be classified as a cooperative game, conditions for cooperation have to be provided [CEW11]. In many multi agent environments these conditions are not guaranteed and therefore agents do not cooperate. An example of such a non-cooperative problem is the prisoner dilemma, where two agents in separate rooms are accused of a crime. They therefore have to confess or keep quiet which results in a non-cooperative behaviour where both confess and go to jail, which can also be described as a social dilemma. If both would have kept quiet they would be free but they choose to confess instead. In this example, the agents would have benefited from a collaboration in which both opted for silence. This raises the question of how such cooperation can be established in a non-cooperative environment. One solution to this problem would be to enable communication between agents. An example of such communication provision was proposed in [SBGP18], where agents were able to trade with each other. By extending the actions of the agents, they were able to communicate and make an offer to the others. This offer consisted of an action, which the other agents could perform and therefore would be given reward by the offering agent. With such trade mechanism, called action trading, a non-cooperative environment can be modified into a cooperative one. This action trading has proved successful, as it has been used in different non-cooperative environments. With the extension of action trading the agents were able to overcome the social dilemma in the environments Iterated Matrix Game and the Coin Game [SBGP18] and therefore were able to outperform selfish agents. As action trading has shown good results, we want to investigate the nature of the agents

with this cooperative approach further. In particular we will be examine the behaviour of agents with action trading regarding the length of the cooperation and additionally we will investigate the impact of some factors on the cooperation of agents.

1.1 Scope of this Work

Game theory provides insight in the cooperative behaviour of agents in a multi-agent environment. Examine the outcomes of a given game with a learned solution concept will give us more information about cooperation between agents. Therefore we will modify the action trading [SBGP18] concept and investigate the cooperative behaviour further. For this purpose we will take a closer look at the scalability aspect of action trading and different factors which could influence the cooperation between agents. Therefore we want to scale the trade related steps from 1 to $n \in \mathbb{N}$ steps. This allows us to compare short- and long-term cooperation between agents. Furthermore we want to alter action trading, where the trade related actions are not necessarily connected to the immediately gain of reward but rather with the path to it. Therefore forcing agents to plan their actions and offers in advance, allowing long-term offers to have more impact. To investigate cooperativeness further we will change different factors related to trades: mark up of compensation, where we increase the amount of reward agents pay for actions. trading budget, which limits the agents trading possibility. By comparing the variations of action trading we want to answer the question if long-term trades improve or lower the rewards of agents over the short-term trades. Another key component of cooperation is the payment timing. In [SBGP18] the offering agent was able to condition the other agent to perform the actions without a trade which was caused by the payment timing. By paying the agent after the actions were executed, the agent had to trust the paying agent to reward him. While this had benefit for both agents at first, soon trust issues emerged as the agent did not pay anymore. To address this problem we want to compare the payment timing before and after the trade related actions were executed. Therefore allowing us to investigate the cooperative behaviour with action trading in regards to trust. Since agents want to maximize their own immediate reward the offering agent has to believe the acting agent that he will fulfil his part of the trade, if the payment timing is beforehand. On the other hand if the payment is executed afterwards, the acting agent has to believe the offering agent to pay. Comparing the two payment timings will answer whether the offering or the acting agent should trust the other agent more. Additionally we will examine the short- and long-term trading behaviour of agents by given them the possibility to decide for themselves how long they want to follow an offer. This will provide us with a more direct insight on the willingness of agents to cooperate short- vs long-term.

2 Related Work

2.1 Action Trading

A recent concept for cooperativeness of agents in a multi agent system is action trading [SBGP18]. With this approach agents were given the possibility to trade with each other, therefore forming cooperative behaviour. These trades were realised by extending the action space of agents, with whom they were able to make an offer to the other agents. These offers consisted of actions, which the other agent could follow and be rewarded with a constant amount by the offering agent. Moreover the offer and supply had to match at the same time step. The action trading concept has shown to be successful, as the agents cooperated with each other. This Cooperation between agents helped to improve the overall alongside their own rewards over the non-cooperative counterpart. The cooperative approach was evaluated in two different domains: Iterated Matrix Game and the Coin Game. In the Iterated Matrix Game two agents had to chose between action 1 or 2. Much like the infamous prisoner dilemma the payoff from the actions heavily depended on the other agent, which resulted in poor rewards due to selfish behaviour. With action trading agents were able to suggest an action in addition to their choice of action, allowing cooperation between the agents. This concept did very well and constantly increased the overall reward during the training process of the agents. In the Coin Game two agents had to collect colored coins in a 2×2 Grid. The agents and the coins were color coded (blue and yellow), which indicated their belongings. For every coin an agent collected he received the same amount of reward. But if the coin does not matched their color of the agent, the other agent received a penalty. Action trading enabled the agents to trade their reward for collecting the coin before the other. While the agents without action trading performed poorly due to their selfish nature, the cooperation with action trading showed agents doing increasingly well.

Open Questions

While the action trading in [SBGP18] has shown decent results, in the Coin Game the amount of trades decreased and continued to doing so after a period of time. The reason for this behaviour was proposed as a result of conditioning, where agents did not offer any trade in order to gain short term reward. Furthermore the action trading was only performed with one action as part of the trade, which opens the question of the scalability of this concept. Additionally the compensation in [SBGP18] was fixed, which raises the question of an optimal amount of compensation.

3 Background

Multi Agent Environments are a great platform for Game Theory to examine. The environment can be used as a mathematical representation of a game and therefore allows us to analyse the behaviour of agents as they interact with the environment. Every agent can be seen as an individual who takes influence on his surroundings. In our case we want to compare the performance of agents in a non-cooperative and a cooperative environment and examine the social behaviour.

3.1 Reinforcement Learning

As Agents interact with the environment, they are only interested in their own goals. Thus we see them as individuals, who act selfish in regard to their reward. Every agent receives observations which can be interpreted as a state of the environment. Since agents are a part of the environment, they change the state by interacting with it. As we want to maximize the reward of the agents by learning, we model this problem as an Markov decision process (MDP). We can use the MDP as a mathematically framework for an agent to learn what action to perform. In our case we have multiple agents and therefore extend the MDP to a Stochastic Game (SG), also referred to as Markov Game [Lit94], where every transition of an state depends on the actions of all agents. Therefore the SG is denoted as a tuple (S, N, A, T, R), where S is a finite set of states, N the finite set of agents, A the joint-actions of the agents, T the transition function and R the reward function. Furthermore the joint-actions A can be described as $A = A_1 \times ... \times Ai$, where A_i is the finite action set of the agent $i \in N$. The transition function $T : S \times A \times ... \times A_i \to R$ indicates the transitions of a given state S with the actions of all agents $A \times ... \times A_i$ into a new state S with the reward R. The reward function $R : S \times A \times .. \times A_i \to R$ describes the reward for every agent, where they perform the actions $A \times ... \times A_i$ in a state S. As we are interested in maximizing the reward, we are going to train the agents with reinforcement learning (RL) [SB98]. There every agent learns a policy $\pi : A \times S \to [0, 1]$, which determines the probability of the agent in a state S to take an action A. Since the agents goal is to maximize the reward, he will optimize the policy π in such way that it leads to the most profitable return. The expected return at the time step t is denoted as $R(t) = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$, where $\gamma \in [0, 1]$ is the discount factor, which represents the importance of future rewards R_{t+1}. To learn the most promising policy π agents we use Q-Learning [SB98], where the state-action function Q is given as $Q : S \times A \to R$. The state-action function Q indicates the value of an action A in a given state S. Therefore the agents can adjust their policy π according to the Q values of a state. While at the beginning the Q values are initialized randomly, they are being updated over time. By selecting an action $a \in A$ at a time step t the agent can observe the reward r from the action which led from state s to s' and update the Q value according to: $Q^{new}(s, a) \leftarrow Q^{old}(s, a) + \alpha(r + \gamma max_{a' \in A} Q(s', a') - Q^{old}(s, a))$, where $Q^{old}(s, a)$ is the old Q value,

$\alpha \in (0, 1]$ the learning rate, r the reward, γ the discount factor and $max_{a'}Q(s', a')$ the expected optimal future Q value. While the individual Q values of every agent are being updated by interaction with the environment, they have to find the policy π in a proper time period. Thus we have to balance the exploration and exploitation process of the learning model. To achieve sufficient exploration we use the decaying ϵ-greedy strategy, where an agent chooses an random action with the probability ϵ and the most promising action $argmax_{a \in A}Q(s, a)$ with the probability $1-\epsilon$. As time progresses we want to decay ϵ with ϵ^{decay} and let it converge to a minimum ϵ^{min}. Therefore the agent will explore more at the beginning of the learning process while slowly adapt the Q values and the policy π. Once ϵ reaches ϵ^{min} we keep exploration at a minimum and can exploit the learned policy π which results in the estimated optimal reward for the agent. Additionally we combine Q-Learning with Deep Neural Networks (DNN) to Deep Q-Networks (DQN) [MKS+15], which are used as a function approximation to represent the state-action function Q. Thus we can represent every agent as a Deep Q-Network allowing them to learn faster as they can use their earlier experiences to predict the outcome in a not visited state. Using experience replay we update the Q values of the Deep Q-Network by randomly selecting previous experienced transitions from a memory buffer. By using two different Deep Q-Networks for an agent we can stabilize the learning process, as we use a policy network and a target network. As the state-action function Q is updated, we need to adjust both sides which results in a unstable divergent learning process. To prevent this issue we use a target network, which maintains the Q values. While we update the policy network constantly we do not update the target network. After a target update period we synchronize both networks and obtain the error of Q to adjust the values accordingly. This will lead to a more stable learning process and therefore reach the optimal policy π faster.

3.2 Cooperative Game Theory

In Game Theory games are represented as mathematical models, in which entities make decision based on rationalism. Those models are used to understand and predict the outcomes of games at a rational level. Games can be divided into two distinct subcategories, non-cooperative and cooperative games. In artificial intelligence research the non-cooperative game (chess, go, ...) has received a lot of interest where the learning players find the most effective way to solve their given problems. In Multi-agent Systems on the other hand, the selfish behaviour of learning agents are a main concern, where they perform poorly due to not cooperating with each other. To address this issue we transform an non-cooperative multi-agent system, the smart factory, in a cooperative one, therefore allowing us to apply game theory of cooperative games. In [CEW11] the authors examined the computational aspects of cooperative game theory, where the solution concept σ is denoted as a function: $\sigma : \Gamma \to 2^{\Omega}$, where Γ a class of games with outcomes Ω. With this function the following problems emerge as state in [CEW11]: *Non-emptiness*, *Membership* and *Computation*. *Non-emptiness* questions if there is any existing rational outcome matching the solution concept. If there is an outcome *Membership* makes the decision if the outcome is rational or not according to the solution concept. *Computation* provides information if an rational outcome can be computed. In multi-agent systems these problems can be addressed: By learning an optimal policy π the agent answer the

question whether or not there is an rational outcome (*Non-emptiness*) and if an rational outcome can be computed (*Computation*). Since agents act according to their learned policy, every outcome is rational (*Membership*) in regards to the solution concept. With reinforcement learning agents learn the solution concept on their own and therefore we are more interested in the outcomes Ω given a game. We therefore want to examine the outcomes and the cooperation between agents, who are given a cooperative game and find the solution concept on their own.

4 Environment

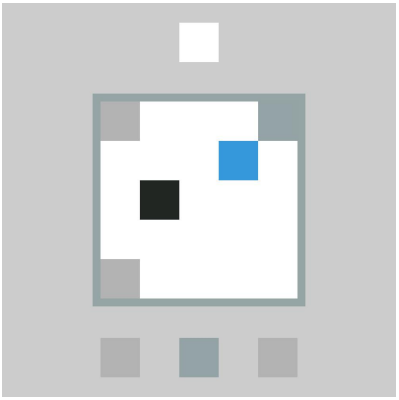


Figure 4.1: Smart Factory: Agents (blue, black) have to process machines in the corners according to their tasks which are shown at the bottom. Both agents have different priorities, displayed at the top, and therefore different rewards.

The environment we use to evaluate the $n \in \mathbb{N}$ step action trading is the smart factory. The smart factory is a 5×5 Grid in which agents interact with their surroundings. In this multi agent environment agents have to execute different tasks, each consist of one machine which has to be processed. The agents will be given three randomly generated tasks and have to execute those sequentially to complete them. There are two different types of machines, which will all be placed in corners of the smart factory. Additionally the machines will become inactive for a fixed amount of time steps once an agent have processed it while it was active. The order of the task and the lack of resources will create competition between agents and motivate them to interact with each other. The possibility to trade encourages the agents to cooperate and allow us to investigate this behaviour. At the start of every episode agents will be spawn on a different location to find the optimal strategy to complete the tasks rather than use the same actions over again. By choosing from a given action set agents can move in four directions: up, down, left and right. Every time an agent performs a movement he will receive a penalty in order to find the fastest way to achieve his objective. The amount of penalty every agent receives is different, which will lead in more cooperativeness. If an agent position matches with the position of a active task related machine, the agent will receive a reward and the machine will become inactive. The reward an agent obtains will also differ from each agent. Those asymmetrical conditions will be created by different priorities. Agents with higher priority will be punished with more penalty but will get more reward from machines. At the beginning of every time step agents receive the state of the environment in form of an observation which also contains the priorities. The observations consisting of multiple layers include information about active machines, agent positions, priorities, tasks and whether or not the other agent has completed his tasks. While in single

agent environments only the agent had influence on the surrounding, in an multi agents environment every agent is part of the changing process. This makes it impossible for agents to perfectly predict the outcomes of their actions. This is where cooperative behaviour of agents can improve the uncertainty by working together for a certain amount of steps. Therefore the competition between agents for machines can create a win-win situation for both by cooperating.

4.1 Tasks

The goal of every agent in the environment is completing all tasks. At the beginning of every episode the agents are given three randomly generated tasks, which have to be executed in order. Every task consist of a machine type which has to be processed. In the observation every task is shown to the agent and the next task also to the other agents.

4.2 Machines

In the environment are two different types of machines. Those are placed in three different corners of the environment and remain their position throughout the episode. At the start of every episode the machines are all active and are visible by all agents. Once an agent reaches the machine and the next tasks of his is the same machine type, the machine will be processed. If processed the machine becomes inactive for 10 time steps. During those time steps the machine is invisible for the agents. When the inactive phase ends, the machine becomes active again until it is processed again. Every time the machine is processed the agent receives reward for doing so.

4.3 Actions

At the start of every time step the agent have a set of actions they can choose from. Depending on enabling action trading and the trading modes the actions may vary. For the basic actions (movement actions) the environment has four different directions an agent is able to go: up, down, left and right. If action trading is enabled the actions are extended by actions with movement and offer parts. Every agent has to choose an action if they did not completed their tasks yet. By taking an movement action they receive a penalty which should motivate the agents to execute the goal as fast as possible. The penalty vary depending on the agent's priority.

4.4 Priorities, Rewards and Penalties

Every agent has a different priority, which influences the rewards and penalty. Every episode the priorities are randomly distributed to the agents. There are two different types of priorities, high and low, as we only have two agents in our environment. With high priority the agent receives more penalty for taking a step, while the low priority agent gets less. This asymmetrical distribution of reward motivates the agents to cooperate with each other, since the high priority agent wants to end the episode faster than the low priority agent.

4.5 Observation

Choosing an action requires the agents to gain information about the state of the environment. Therefore we use observations which are a simplified view of the environment. For the basic observation with no action trading we display nine different channels. Every channel shows the 5×5 Grid of the smart factory where the cells are filled with either ones or zeros. At the start of every time step the agents receive their observation of the environment. The first channel represents the active machines, where the associated cells are filled with ones. The agent's own position is shown in the second channel. The third representing the own priority by either filling all cells with ones for high priority or zeros for low priority. The own tasks of the agents are represented in the channels four, five and six. The other agents position is displayed in the seventh channel, with the next task of him shown in the eighth. To indicate if the other agent already completed all tasks we use the ninth channel. While the observation does include the sufficient information of the environment, we will later add more channels to help the communication between agents.

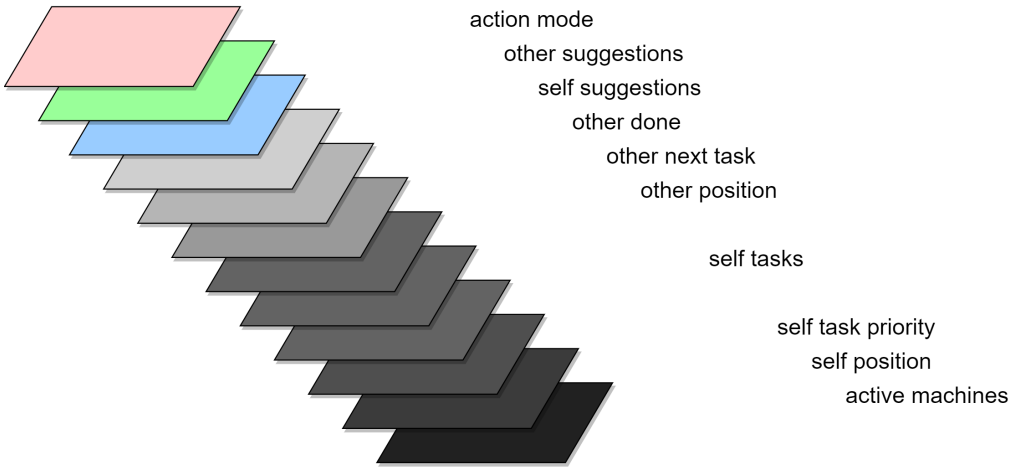


Figure 4.2: Shown are all observation channels of the observation. While the lower nine channels represent the information of the environment, the channels *self suggestions*, *other suggestions* and *action mode* are dedicated to action trading between agents.

5 Action Trading

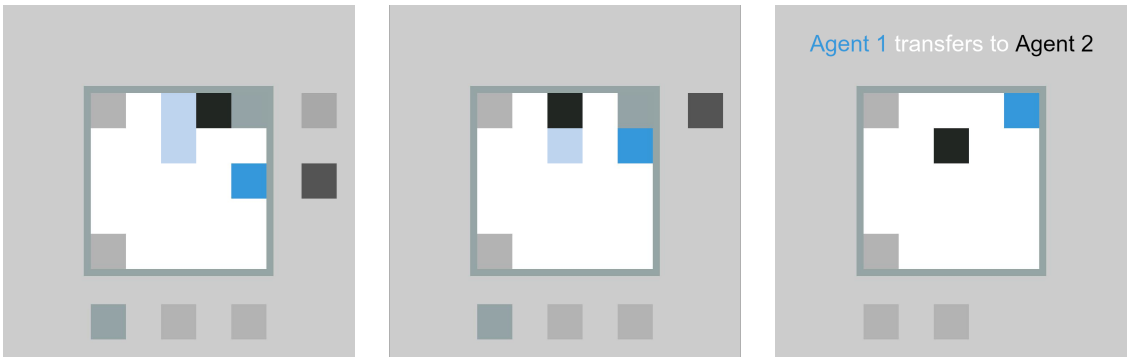


Figure 5.1: Example of Action Trading with two offer actions: Agent 1 (blue) makes an offer "move left and down" to Agent 2 (black) at time step t (left). Agent 2 follows the offer at $t+1$ and $t+2$. Agent 1 compensates Agent 2 for completing his offer at $t+2$ (right).

Many Multi-Agent Systems are modeled for competitive purposes, where agents compete for resources against each other. Every agent tries to maximize their own reward and therefore act selfish and do not care about other agents. The smart-factory is not different and therefore does not provide any cooperative interactions between agents. To transform the non-cooperative environment smart-factory into a cooperative one, we extend the action space of the agents. By extending the actions, the agents gain the ability to interact and cooperate with each other. Specifically we use a modified version of action trading [SBGP18] where agents can trade their reward for actions performed by the other agent. This cooperative actions have already shown good result in the domains Iterated Matrix Game and the Coin Game, where agents improved the overall reward alongside their own over their non-cooperative counterpart. We therefore want to take this concept and apply it to our environment smart factory. In [SBGP18] action trading was performed with one action as part of the trade, which were shown to be successful. We want to scale this approach even further to n offer actions. To enable the multiple actions as part of the trade the offer has to be separated from the supply. We therefore modify the environment step by appending the *action comparison* and the *add offer* functions. This allows the agents to trade with each other and let us track the cooperative behaviour of the agents. While in [SBGP18] the amount of reward the following agent received was fixed, we want to calculate the compensation depending on the state of the environment and actions of the offer. Therefore the trade becomes more fair and thus motivate the agents further to cooperate. Additionally we introduce a different trading mode, which will keep the amount of actions the agents can choose from at a constant. This enables us to compare the offers with various amount of offer actions more directly and therefore the short- and the long-term cooperativeness.

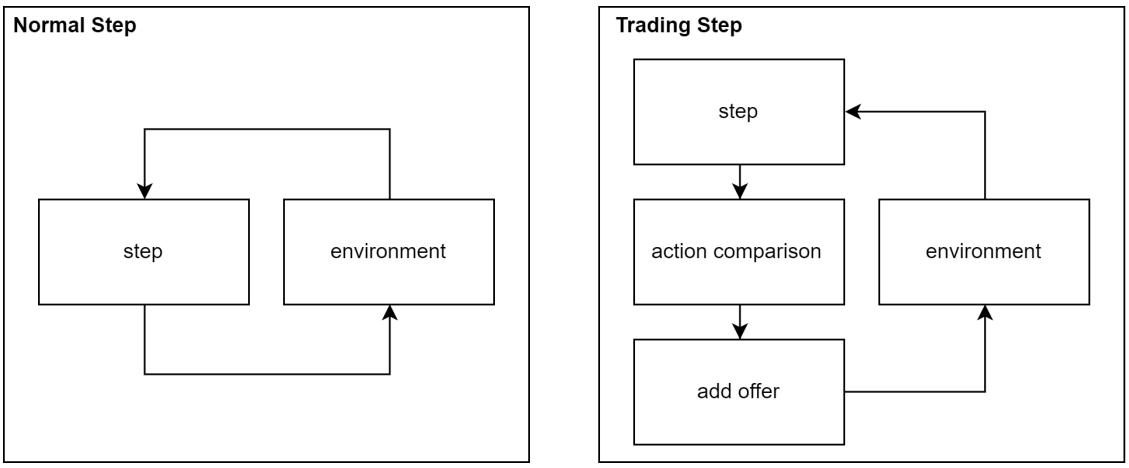


Figure 5.2: Trading Step: Using action trading requires modification on the step of the environment. By extending the normal step on the left with two functions *action comparison* and *add offer* agents are able to trade with each other. This extended step is called *trading step*.

5.1 Modification on Action Trading

While in the action trading [SBGP18] the trade consisted of one action and a fixed amount of compensation, we want to take the concept and scale it to $n \in \mathbb{N}$ offer actions. Extending the number of offer actions will require some changes on the original action trading. While in [SBGP18] the offer and supply matched at the same time step t, in our case we want the offering agent to suggest an offer at time step t to the other agent, who can then decide whether to follow the offer or not at the next time step $t + 1$. By delaying the supply part of the trade we will be able to extend the offer to n numbers of actions, where every action of the following agent has to match with the offer action at time step $t + n$. To achieve this separation of offer and supply we introduce two different functions, *action comparison* and *add offer*. With *add offer* the agents are able to suggest an offer, consisting of n offer actions, to an other agent at time step t. *Action comparison* allows us to compare the current action the agent performed with the offer action that was suggested to him at any given time step $t + n$. Once all the offer actions has been followed the agent gets compensated by the offering agent. In the tested environments of [SBGP18] the amount of reward the agent got for following the offer was fixed. We want to change the compensation to fit our needs in the smart factory. Therefore we calculate the compensation at time step t, where the offer is made. Depending on the state of the environment and expected reward of the agents the compensation may vary for every time step. This will make the trade more fair for both parties. The following agent only gets paid if he executed all the n offer actions at every time step $t + n$ with the compensation. If any of his actions at time step $t + n$ varies from the offer action the offer will be withdrawn. Additionally every agent is only able to make one offer at a time.

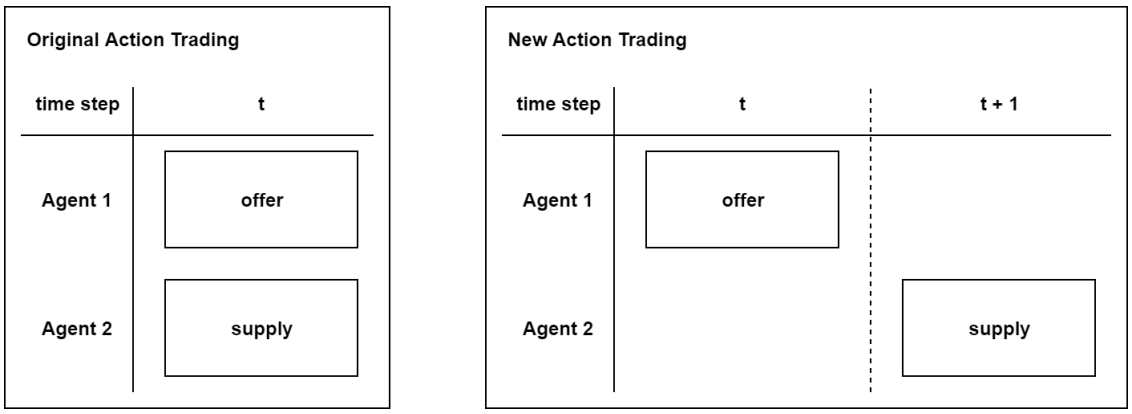


Figure 5.3: Split of offer and supply in Action Trading. On the left original action trading [SBGP18] the offer and supply had to match on the same time step. The new action trading on the right split offer and supply in to part where the supply can only match one time step after the offer was given.

5.1.1 Action Comparison

If the agent is given any offer, we compare the current action of the agent with the offer action in the *action comparison* function. At the start of the time step the agent picks an action, which will be stored in a log, and takes the step in the environment according to the chosen action. Once the step is executed the *action comparison* takes the last element of the log which is the current action. Provided the agent has been offered any actions, we compare the current action with the first action of the offer. If the two match, the first offer action will be deleted. This process repeats for the n offer actions, where every time step only one offer action will be compared. Once every action in the environment matched with the offer actions, the offering agent has to pay the following agent. If at any time step the action of the agent and the offer action do not match, the entire offer will be canceled and the following agent will receive no compensation at all. If the offering agent has sufficient money to pay the following agent, they agents exchange the reward and therefore the trade is successful. Once the trade is completed or the agent has no offer suggested to him, the other agent is able to make an new offer again.

```
ACTION COMPARISON

for agent in agents:
    if offer exists:
        if current action == offer action:
            delete offer action from offer
            if no more offer actions:
                pay reward
        else:
            delete offer
```

5.1.2 Add Offer

Every time step agents have the opportunity to make an offer to the other agent. At the beginning of every step in the environment agents have to select one of the given actions. If the chosen action includes an offer, the other agent will be given the proposal, if there is none already existing. Depending on the trading mode, agents chose the offer at the step in the environment or add the offer actions step by step. Once the agents have an complete offer to make, the proposed actions will be represented as a queue and also displayed in the observation of the environment to both agents.

```
ADD OFFER

for agent in agents:
    if no offer:
        if new offer:
            add new offer
            calculate compensation
            add compensation
```

5.2 Trading Modes

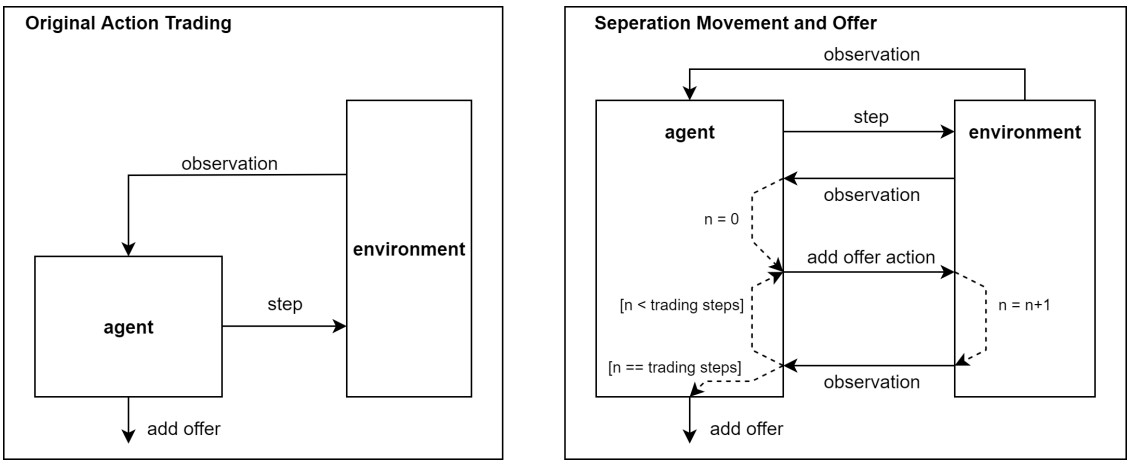


Figure 5.4: Shown are the processes of adding offers for the different trading modes, *Original Action Trading* and *Separation Movement and Offer*. While agents with *Original Action Trading* add the offer by selecting one action, agents with *Separation Movement and Offer* have to pick every offer action separately.

In the original action trading [SBGP18] the concept of action trading was implemented by a cross product of movement and offer. There agents chose one single action to move and make an proposal. This worked well for one trading step, but since we want to scale this approach is becomes rather complicated since every additional step increases the action space of the agents exponentially. As the action space increases, it becomes more and more difficult for agents to find the optimal policy π. Therefore we are going

to introduce a modify version of the original action trading where the movement and the trading part are chosen separately and thus reducing the action space. To evaluate the new concept we will compare both while scaling the trading steps.

5.2.1 Original Action Trading

The *Original Action Trading* unites the movement M and the offer O in one single action. The action space A, every action consisting of a tuple, can be expressed as a cross product: $A = A_M \times {A_O}^n$. Where $n \in \mathbb{N}$ is the amount of offer actions every agent can propose to the other. Given the different tuples in the environment, the amount of actions the agents can choose from is:

$$|A| = \begin{cases} |A_M| & \text{for n} = 0 \\ |A_M| + |A_M|^{n+1} & \text{for n} > 0 \end{cases}$$

Where agents can decide if they want to make an offer or not. Thus every agent has to learn from the action space which grows exponentially with the amount of n offer actions. While this concept will lead to good results for low n, it becomes more and more difficult for higher n.

5.2.1.1 Execution

At the start of every step agents choose from a given action set, consisting of the previous mentioned cross product. Once an agent has picked one of the actions, he moves in the environment and makes a proposal to the other agent accordingly to the action. Therefore agents only have to pick one action at any step, but have to learn from a huge variety of actions.

5.2.1.2 Observation

While the normal observation of the environment only shows information of the smart factory, we also want to display the trade related offer actions of the agents. Therefore we introduce two new channels, which will contain all the offer actions an agent received and the offer the agent has made. This allows the agents to predict to what state the offer of the other agent will lead to and also make it easier to decide whether or not the proposal has any meaningful positive impact on their reward.

5.2.2 Separation Movement and Offer

To solve the problem of the exponentially growing action space of the *Original Action Trading*, we introduce a new variation of action trading, where we split the selection of the movement and the offer. This allows us to keep the action space at a constant number while also make it possible to scale the amount of trading steps $n \in \mathbb{N}$ indefinitely. Therefore the action set A of the smart factory can be expressed as: $A = (0, 1, 0), (0, -1, 0), (-1, 0, 0), (1, 0, 0), (0, 1, 1), (0, -1, 1), (-1, 0, 1), (1, 0, 1)$. Where the first two element of every element refer to the movement and the last element indicating if the agent wants to make an offer 1 or not 0. At the start of every step the agents select an action, where they signalize if they want to make an offer or not. If they decide to make

one, they have to pick actions n times as the offer consist of n offer actions. Every time the agents selects an action, the memory gets a new entry. As the agents learn from the memory, they also learn what offer actions to suggest. Once the agent has picked all the n actions, the offer will be proposed to the other agent. The limited action space will enable the agents to learn faster, compared to the *Original Action Trading*. While they do so, they also have to learn to plan ahead, as the offer consists of a sequence of action selections.

5.2.2.1 Execution

By choosing an action at the beginning of every step, the agent has the possibility to propose an offer to the other agent. After the agent decided on an action, he will move in the smart factory to the location. If the agent wants to make an offer, indicated by the last element of the chosen action, he has to select actions in the *add offer* part of the trading step. There the agent receive a new observation, which not only contains the information of the smart factory and the offers, but also indicates that the next action he will pick is being interpreted as an offer action. Once the agent selected an action, it will be appended to the proposal and he has to choose actions again depending on the new observation until the n offer actions of the offer are completely added. Every time the agent picks an action, he will learn from his decision, since the previous observation of the environment, the selected action and the afterward observation are stored in his memory. Once the offer is complete the proposal will be given to the other agent. By splitting the selection of movement and offer, the number of actions remains unchanged for any amount of trading steps $n \in \mathbb{N}$. Hence agents do not have to deal with an exponentially growing action set and are able to learn the optimal actions faster than in the *Original Action Trading*.

5.2.2.2 Extra Observation Channel

The observations of the environment in the *Original Action Trading* 5.2.1 already contain the offers from both agents. But the agents in *Separation Movement and Offer* have to know whether the next action they choose is a movement or an offer action. By adding an extra observation channel the agents will receive the information about their next action. Zeros in the channel indicates that the agent has to take a movement and a decision to make an offer or not. Ones in the channel shows the agent that the next action they pick will lead to a offer action for the other agent.

5.3 Compensation

As agents act selfish and want to maximize their own reward, they have to be compensated appropriately for helping others and sacrifice their immediate reward. Where action trading only enables agents to cooperate with each other, they still have to be motivated to do so by gaining reward. In our environment smart factory agents compete for the limited resource machine while also receiving high penalty for unnecessary waiting. Rather than acting selfish and try to get to the machine as fast as possibly with action trading they can trade their reward for a sequence of steps, which will result in a win-win situation for both agents. While one agent gets to process the machine and only has to pay a small price to

the other agent, the receiver gets compensated with even or more reward he had initially expected for competing with the payer. Hence the questions rises how high the amount of reward has to be for the receiver to be compensated. Therefore the expected rewards for taking an action plays a huge role in this discussion, since agents in a given state select their actions by comparing the expected rewards of the available actions and choosing the most profitable action. Those expected rewards for an action a in a state s under the policy π and following the policy afterwards is denoted as $Q^{\pi}(s,a) = E[\sum_{t=0}^{\infty} \gamma^t R_{t+1}|s,a]$, where the rewards in the future is discounted by γ. To make an action a' as viable as the most promising action a_{max}, the expected reward has to be the same. Thus to make the decision on choosing an action a' over the action a_{max} equal or even better, the expected rewards has to be the same or higher: $Q^{\pi}(s,a') + C \geq Q^{\pi}(s,a_{max})$, where C is the compensation for taking the action a'. The difference D between the two actions a_{max} and a' is therefore $D = Q^{\pi}(s,a_{max}) - Q^{\pi}(s,a')$. The expected rewards $Q^{\pi}(s,a)$ can also be written as: $Q^{\pi}(s_0,a_0) = R(s_0,a_0) + \gamma R(s_1,\pi_1) + ... + \gamma^t R(s_{t+1},\pi_{t+1})$, where the reward of the t-th step in the future is discounted by γ. Since in action trading the agent gets compensated at the n-th step in the future, we have to adjust the discounted expected future reward by $1/\gamma^n$, giving us the total compensation of:

$$\text{Compensation: C } = M_c * \sum_{l=0}^{n-1} \frac{Q_{max}(t+l) - Q_{offer}(t+l)}{\gamma} \tag{5.1}$$

Where Q_{max} is the highest expected reward of the actions in a state s, Q_{offer} the expected reward of the suggested step in the state s, n the amount of trading steps, t the current time step, l the l-th step into the future and M_c an mark up to motivate the following of the offer.

While the agent's expected rewards of every action in every state vary throughout the learning process, we have to find a more stable solution rather than taking the ones of the learning agents. Therefore we train agents without the action trading possibility and use the trained agent weights to create a valuation net for the compensation. This allows us to have the right expected rewards of 5.1 and thus the right value for the compensation.

5.3.1 Execution

An offer in action trading consist of two components, the offer actions and the compensation. While the offer related actions are chosen by the agents, the compensation of the offer has to be calculated according to the offer actions. The offer actions are modeled as a queue and are therefore easily accessible. Getting the right compensation first requires the environment to be stored, since we want to reset to this point after the calculation is done. After the state has been saved, we calculate the compensation of the current state by getting the expected rewards of the following agent. From the expected rewards we get the highest element and subtract from it the first offer actions expected reward. After that we divide the difference by the discount factor γ and add it to our full compensation of the offer. Next we perform a step in the environment by letting the following agent execute the first offer action which is stored in the first element of the queue. Meanwhile the offering agent chooses an action depending on his policy. This step result in a next state with new expected rewards, where we once again extract the difference from the expected rewards of the following agent, divide it by the discount factor γ and add it

to the full compensation. We repeat this procedure for $n-1$ steps of the offer, since we got the compensation of the first action beforehand, which results in the complete compensation of the offer. To motivate the trade for the potential receiver we multiple the compensation with a mark up value M_c and get the amount of reward from 5.1.

```
CALCULATE COMPENSATION

save environment state

compensation = (Q_max − Q_offer)/γ

for t in n−1:
    action_receiver = offer action (t)
    action_payer = action_π
    step(action_receiver, action_payer)
    compensation += (Q_max(t) − Q_offer(t))/γ

load environment state

return compensation
```

5.3.2 Payment Timing

In the original action trading [SBGP18] the trade was being performed in one single time step t, where the offer matched the supply. For any successful trade, the payment between agents was also executed at the same time step t. In our case we can not exchange the compensation at the same time step where offer and supply match, since we split both into different steps. Thus the question of payment timing arises. We will compare the two possibilities for the exchange of compensation; before and after the offer actions were executed.

5.3.2.1 Payment After

Paying the receiver after the offer actions has been executed results in a positive mindset for the offering agent, since the agent only has to pay the compensation after his request has been fulfilled. Hence the agent can only profit from making an offer, as the agent only loses reward if the trade has been completed, which would result in a gain of reward anyway.

5.3.2.2 Payment Before

Exchanging the compensation before the execution of the offer actions results in a more mindful decision to make an offer. By paying the receiver at the same time step as the offer proposal, the agent loses reward before seeing any benefit. Therefore the agent has to decide whether he wants to trade his immediate reward for a future uncertain gain or just keep it. Nevertheless they will get their payed amount of compensation back if the offer has not been completed.

5.3.3 Partial Trading

Until now the agents got only payed if they had executed all the offer actions. Therefore the following agent has to make the decision to accept the offer and follow all of the offer actions or decline it. This acceptance heavily depended on the current state of the environment, as the agent has to decide at this time step to follow or not. But as the agents move and alter the environment, the interest in following the offer may vary as well, since in the new state the offer could not appeal to the agent anymore. Hence the agent decides to dismiss the offer and execute the most promising actions instead. This results in a partial fulfillment of the offer, where the agent get not payed at all. To improve the action trading and take the uncertainty of the environment into consideration we introduce partial trading. With partial trading the agents are able to only perform i actions of the total n offer actions and still get payed for executing those. Since the state of the environment has a huge impact on the following of the offer, the agent does not have to sacrifice any reward by only executing partial offer actions. This will allow agents to dynamically decide on the amount of offer actions they want to follow.

5.4 Parameters

While we want to scale the action trading concept to $n \in \mathbb{N}$ offer actions, we are also interested in finding the optimal amount of offer actions. To do so we will vary trade related parameters such as mark up M_c or budget B. This will allow us to further analyse the social behaviour of the agents.

5.4.1 Mark-Up

By varying the mark up M_c we want to encourage the agents to offer and follow trades, since the benefit of an offer can only apply if both parties profit from such trade. While the mark up M_c proposed in the compensation part 5.1 was only 1.0, this will only make the decision to follow the offer indifferent to the most promising action. Thus we want to increase the mark up M_c to motivate the agent to follow. On the other hand if the mark up is too high, the proposer will not make any offer, as he will not benefit from the trade anymore. Therefore we are interested in finding the sweet spot for both parties. This optimal mark up will lead to more insight into the agents and their perception of the environment, as well as the cooperative action trading approach.

5.4.2 Budget

Limiting the agents trading possibility by introducing a fixed trading budget B could result in a more careful offer making. Since there is only a fixed amount of reward the agents can trade, they have to schedule their budget mindfully to get the most out of it. On the other hand if there would be a great trade opportunity but no budget left, the agents miss out of potential reward gain. We want to examine the impact of budgeting on the trading agents as we increase the budget and compare the results with each other.

6 Experiments

We use the smart factory as the environment to evaluate the modified version of action trading. For our purpose we use two agents, as we only want to scale the offer related actions rather than the environment. Those agents, each represented by two Deep Q-Network, will have to complete their individual three randomly generated tasks. The training episodes will only be finished if both agents have processed all their tasks or the time steps reach $t = 1000$. Every DQN will receiver a input consisting of the channels of the observation, which will then be passed to two sequential linear layers, the first with 32 nodes and the second with 16. After those two layers we receive an action from the output layer. Those two hidden layer with exponential linear unit activation (ELU) are sufficient for our specific problem, as they can handle the size of our environment. For the optimizer we use the Adam algorithm with a learning rate of $5e^{-4}$. Using the decaying ϵ-greedy approach we start with $\epsilon = 1$ and decay it with $\epsilon^{decay} = 8e^{-6}$ until we reach the minimum of $\epsilon^{min} = 0.01$. As we mainly use 1500 episodes of training for the agents this exploration rate will be adequate. Furthermore we use the learning batch size of 64 and update the target network after 2000 training cycles. For the reinforcement learning part we use the discount rate $\gamma = 0.95$, since the current rewards are only slightly more important than the future rewards and will allow the agents to plan in advance what actions to perform. The tasks in the environment involve machines, of whom there are two types from, which have to be processed in order to gain 1.0 reward. After processing the machines will become inactive for 10 time steps, since this waiting time is long enough for the agents to move across the entire smart factory. While there are three machines in the 5×5 Grid of the environment, they are a limited resource. This lack of machines create competition between the agents, as they are both interested in finishing their tasks as fast as possible to gain the most reward. For every step the agents take, they get punished with a penalty. This penalty will depend on the priority of the agent. The agent with the high priority will lose 0.5 reward, while the low priority one will only lose 0.02. The asymmetrical penalty will motivate the agents to find a cooperative rather than selfish solution to finish their tasks quickly. In our case we will be using action trading as our cooperation between both agents and compare the results of the experiments using the rewards. To evaluate our version of action trading we will first compare the agents rewards in an cooperative and non-cooperative environment.

6.1 Action Trading vs No Action Trading

In this section we compare the non-cooperative approach without action trading with the cooperative concept of action trading. Moreover we use action trading with one action offer to evaluate the performance of action trading. We use the *Original Action Trading* mode as we want to analyse the adaptation of action trading to the smart factory environment. First we set the mark up M_c to 1.0, which allows the direct comparison between action trading and no action trading and create no additional benefit for trading agents.

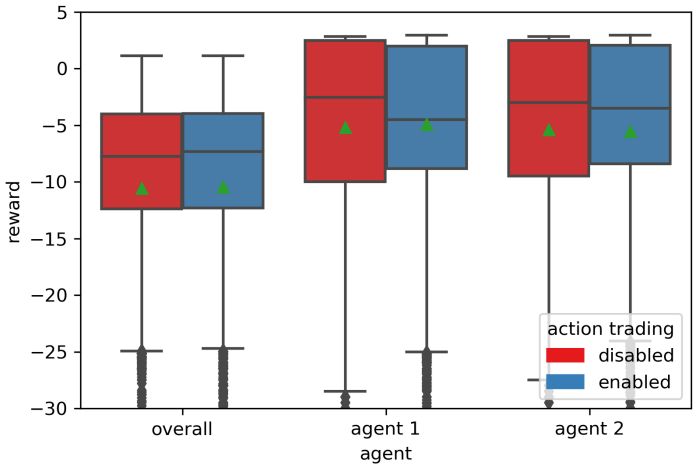


Figure 6.1: Agents with and without action trading: Shown are the rewards of agents with and without the possibility to trade of 20×2000 episodes. Those agents were trained during 20 runs with each consisting of 1500 episodes.

Since the results of the trained agents varied heavily from run to run, we used 20 runs for trading and no trading, each consisting of 1500 episodes to evaluate our approach. After the training phase we used the agents of those runs to perform another 2000 episodes in our environment to create the data of 20×2000 episodes we evaluated. The results are shown in figure 6.1 where we compare the rewards of agents with and without action trading. There we compare the overall reward as well as the individual rewards of every agent. For the comparison we use the arithmetic mean over all runs and episodes $R = \frac{1}{n} \sum_{i=1}^{n} r_i$, where R is the averaged reward over the n episodes of all runs and r_i is the reward of the agents in the n-th episode. The green arrows in 6.1 indicate the arithmetic means of the rewards. Additionally we compare the quartiles and median of the data. Those are shown in the colored boxes, where the 25% quartile is the lower bound, the median the middle line and the 75% quartile the upper bound of the box. The results of the comparison of agents with and without action trading show as slight increase in reward. Those are displayed in figure 6.1, where the mean of the overall reward are $R_{no} = -10.57$ for the non-cooperative behaviour and $R_{tr} = -10.45$ for the cooperative trading approach. Furthermore we can see a marginally higher 25% and 75% quartile as well as a higher median of the rewards from the agents with action trading possibility. Those improvements in rewards are the result of action trading, as the agents completed 2.8 trades on average during one episode. While the difference between the reward of agents with and without action trading seems to be very small due to the mark up $M_c = 1.0$, we have to factor in the difference between the action spaces. For the non-cooperative agents the action space only consisted of 4 actions $A_{no} = ((0,1),(0,-1),(-1,0),(1,0))$, whereas the agents with action trading had to learn 20 actions $A_{tr} = ((0,1,0,0),(0,-1,0,0),...,(1,0,1,0))$. As this is a major factor in the learning process of the agents we can assume that the trading agent outperformed the agents without trading by a margin even though they were handicaped from the mismatching action space. On top the 25% and 75% quartiles

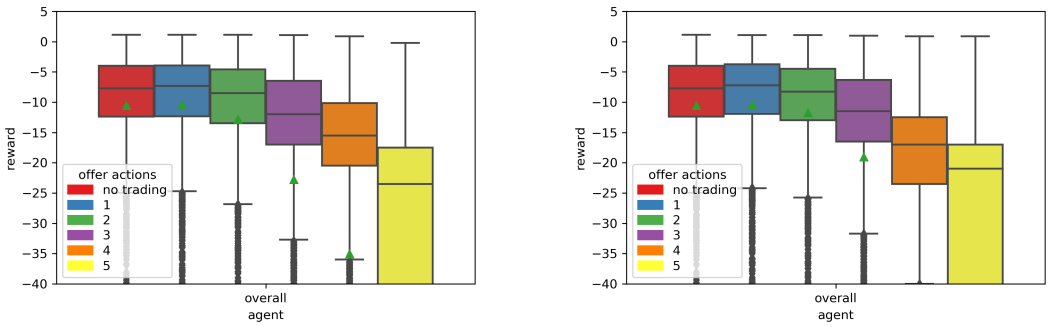
for both individual trading agents were narrower compared to the agents without action trading, as shown in figure 6.1. This may due to the fact, that the agents can make and follow offers in situations where they would otherwise do not expect reward from any movement action. Hence they can follow an offer in order to gain reward regardless.

6.2 Trading Modes

In this section we want to evaluate the scaling aspect of our modified version of action trading. As we already seen improvement in reward for agents with trading over agents without trading, we want to scale the amount of offer actions to $n \in \mathbb{N}$. Furthermore we want to compare both trading modes proposed in 5.2 and verify our assumptions on the *Original Action Trading*. While we examine the trading modes individually, we also take a look at the behaviour of the agents with more mark up M_c. Additionally we will only use $n \in [1, 5]$, as the agents will trade less for higher n and therefore the trades become less important in regard to the reward.

6.2.1 Original Action Trading

The *Original Action Trading* uses a single action to determine the movement and the offer. This leads to an exponentially growing action space which was proposed in 5.2.1. While the original action trading adaptation should show reasonable results for low n, we except worse performance for higher n. To verify this assumption we trained agents with $n \in [1, 5]$ offer actions for 20 runs each. Those runs consisted of 1500 episodes like in 6.1. Our observations are based on 20×2000 episodes in the environment after the training process of the agents. The results with mark up $M_c = 1.0$ in figure 6.2a confirm our assumptions, as the agents rewards already decrease at $n = 2$ and continue to do so for higher n. While agents with action trading achieve more rewards than no trading agents for $n = 1$, they perform worse for $n \geq 2$. The reason for the decreasing reward



(a) Original Action Trading: $M_c = 1.00$

(b) Original Action Trading: $M_c = 1.05$

Figure 6.2: Rewards of Agents with trading mode *Original Action Trading*: For lower n agents are able to outperform no action trading agents. As n increases the reward drops significantly due to the action space. An 5% increase of mark up M_c (right) show overall better results than $M_c = 1.0$ (left).

is due to the varying action spaces of the agents. While agents without action trading have an action set of 4 different actions, the action space of the trading agents increases alongside n. Therefore the action space for agents with action trading $n \in [1, 5]$ strongly correlate with the rewards, which are shown in table 6.1. Alongside the rewards we can see the decrease in completed trades as well. This may due to the lack of learning caused by the expanding action space, but can also indicate to the social behaviour of agents, who do not want to cooperate with each other for a longer period of time.

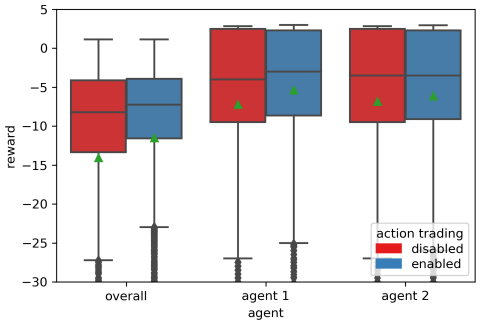
Motivating the cooperative behaviour of agents, we use the mark up M_c of 1.05. While the normal compensation of 5.1 with $M_c = 1.0$ made the decision between the offer following and the most promising actions indifferent, the 5% increase should encourage the following agent to perform the offer actions more. Therefore we assume more trades completed and different reward overall, as the trading behaviour of the agents change due to the increase of compensation. For the comparison between action trading with $M_c = 1.0$ and $M_c = 1.05$ we executed the previous runs again, only this time with the mark up raised. The results in 6.2b show overall higher rewards for agents with mark up $M_c = 1.05$ over $M_c = 1.0$, except for $n = 4$. Thus we can say that the increase of compensation does lead to slightly higher rewards, but not necessarily in more trades, see 6.1. While agents with $n \in [2, 4]$ complete more trades with $M_c = 1.05$ over $M_c = 1.0$, the increase is only marginally. The unexpected drop in reward for $n = 4$ can be explained by the learning process. As the action space increases the performance heavily depend on the exploration of the agents. Since they choose random actions at the beginning of the learning process, their learned actions at the end depend mostly on those initial selection. Therefore the results of the *Original Action Trading* for higher n are mainly affected by randomness and therefore do not suit for analysing the social behaviour of the agents.

n	mark up	action space	trades/episode	mean reward	median reward
-	-	4	-	-10.57	-7.74
1	1.0	20	2.80	-10.45	-7.32
2	1.0	68	0.60	-12.82	-8.5
3	1.0	260	0.13	-22.77	-12.00
4	1.0	1028	0.03	-35.02	-15.50
5	1.0	4100	0.00	-86.71	-23.50
1	1.05	20	2.78	-10.40	-7.20
2	1.05	68	0.62	-11.74	-8.24
3	1.05	260	0.13	-19.03	-11.50
4	1.05	1028	0.03	-49.30	-17.00
5	1.05	4100	0.00	-70.66	-21.00

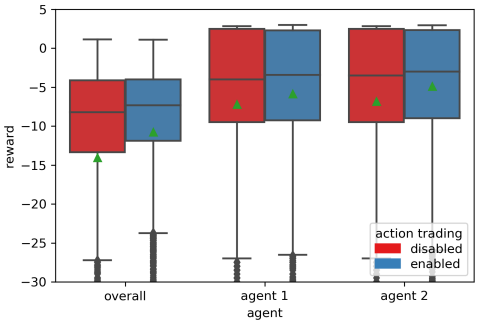
Table 6.1: Scaling of n with *Original Action Trading*: As n increases the action space grows exponentially, while the average trades and the rewards of every episode decrease.

6.2.2 Separation Movement and Offer

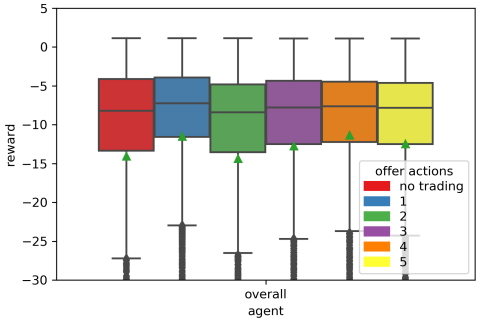
The *Separation Movement and Offer* mode splits the movement and offer selection into two separate parts. This allows the action trading to better scale, as there is not exponential growing action space. The agents can choose from 8 different actions and the action space do not depend on the amount of offer actions $n \in \mathbb{N}$. For easier comparison between agents with and without action trading, we use the same action set for both agents. For the results in 6.3 we trained the agents in 20 runs, each consisting of 1500 episodes, and got the evaluation data from additional 20×2000 episodes. For our first comparison on the agents with $n \in [1, 5]$ offer actions we use the mark up $M_c = 1.0$. The results in figure 6.3a show the agents with $n = 1$ do outperform the agents without trading possibility. But the reward for $n = 2$ are extremely low compared to $n = 1$, see 6.3c. For $n \geq 2$ we see the overall reward increasing until $n = 4$. This sudden drop in reward for $n = 2$ may be caused by the learning process of the agents. As the trading mode *Separation Movement and Offer* stores every choice of actions (movement and offer) in the memory



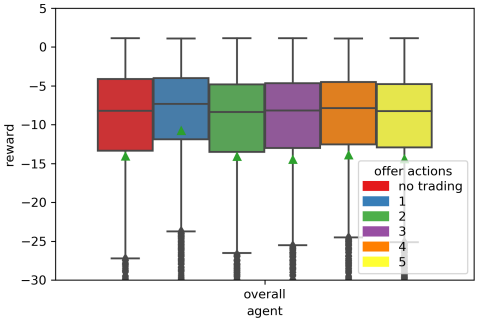
(a) Action Trading vs No Action Trading: Separation Movement and Offer: $M_c = 1.00$



(b) Action Trading vs No Action Trading: Separation Movement and Offer: $M_c = 1.05$



(c) Separation Movement and Offer: $M_c = 1.00$



(d) Separation Movement and Offer: $M_c = 1.05$

Figure 6.3: Rewards of Agents with trading mode *Separation Movement and Offer*: For $n = 1$ agents exceed the reward of the agents without action trading. For $n = 2$ the rewards drop extremely and recover for $n \in [2, 5]$. The increase of mark up to $M_c = 1.05$ (right) decreases the reward for $n = 1$ and also reduces the recover of $n \in [2, 5]$ compare to $M_c = 1.0$ (left).

of the *DQN*, the agents may can not learn to add offer actions properly due to the exponential expansion of the memory entries. While the batch size for the learning agents are constant, they entries in the memory keep growing with n. This make it harder for the agents to learn, since the learning batch the agents receive are selected randomly and at a constant size. Therefore the agents with $n \geq 1$ could already encounter issues due to the batch size limit which causes more random results in our data. However the cooperative behaviour could also be influenced by a different factor. The amount of overall completed trades are shown in table 6.2, where the agents with $n = 1$ trade 1.88 times on average during an episode. For $n \geq 2$ we see the trade amount drop significantly. Thus we can say the agents choose to cooperate for a short amount of time rather than work together for long.

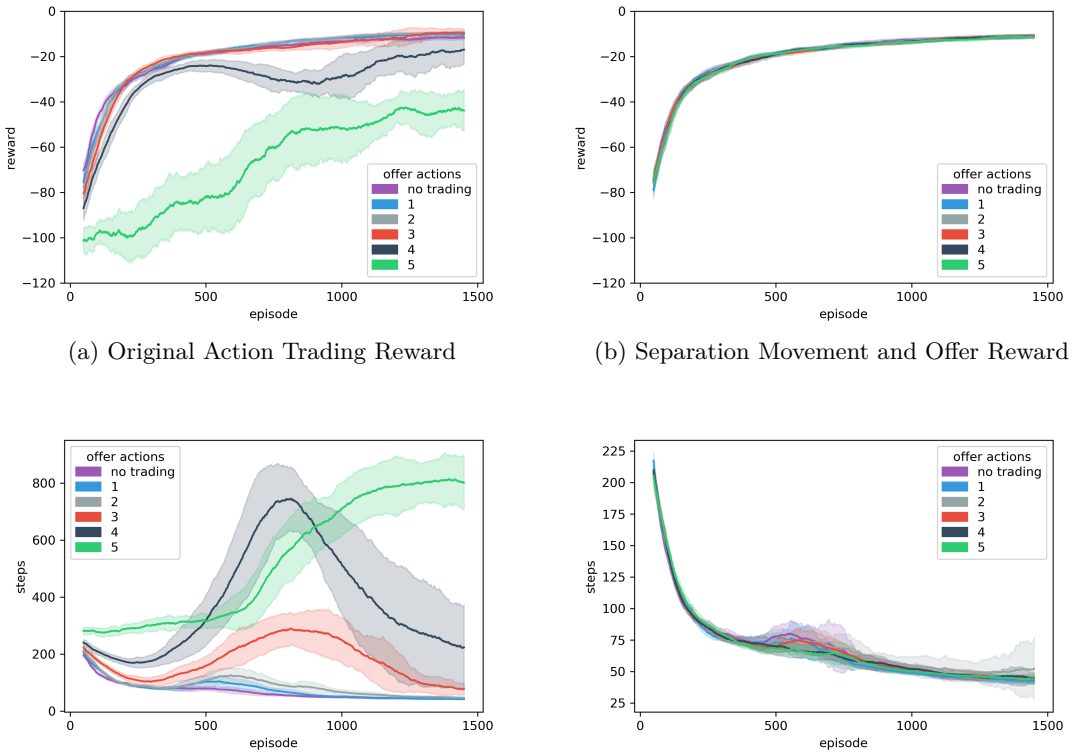
Since we want to encourage the agents to trade, we repeated the runs with the mark up $M_c = 1.05$. This additional compensation serves as a motivation for the following agent. While $M_c = 1.0$ does not provide any benefit for the following agent, with $M_c = 1.05$ the agent expects to gain reward by completing the offer actions. This should be reflected in the trading behaviour of the agents. The results in 6.2 show that agents with $M_c = 1.05$ do not complete more trades for $n = 1$, but they do overall increase for $n \geq 2$. The drop in trades for $n = 1$ causes the agents with $M_c = 1.05$ to perform worse than with $M_c = 1.0$, shown in 6.3b. Also the increase of trades for agents with $M_c = 1.05$ and $n \geq 2$ show worse rewards than for $M_c = 1.0$, see 6.3d. This rather odd behaviour could be explained by the mark up $M_c = 1.05$, as it may be not appealing enough for the agents to gain any real benefit.

n	mark up	action space	trades/episode	mean reward	median reward
-	-	8	-	-14.04	-8.22
1	1.0	8	1.88	-11.49	-7.24
2	1.0	8	0.40	-14.34	-8.4
3	1.0	8	0.16	-12.73	-7.8
4	1.0	8	0.11	-11.33	-7.64
5	1.0	8	0.04	-12.45	-7.84
1	1.05	8	1.70	-10.75	-7.34
2	1.05	8	0.51	-14.07	-8.36
3	1.05	8	0.17	-14.48	-8.16
4	1.05	8	0.09	-13.91	-7.88
5	1.05	8	0.05	-14.41	-8.26

Table 6.2: Scaling of n with *Separation Movement and Offer*: Increasing n keeps the action space constant, while we see a drop in trades. The immediate increase of reward for $n = 1$ drops for $n = 2$ and recovers afterwards. The 5% increase of mark up has mainly negative impact on the rewards.

6.2.3 Comparison between Trading Modes

In this section we want to compare the different trading modes. The *Original Action Trading* is not suited for more offer actions $n \in \mathbb{N}$, since the action space grows exponential with n (see 5.2.1). Therefore we introduced the trading mode *Separation Movement and Offer*, where we keep the action space constant. This should allow us to scale the action trading concept to an unlimited amount of action offers $n \in \mathbb{N}$. In 6.2.1 and 6.2.2 we evaluated the results we gained from the trained agents over a period of 20×2000 episodes. The agents with action trading possibility at $n = 1$ did outperform the non-cooperative counterpart across the results. For higher n the agents with the trading mode *Original Action Trading* performed very poorly, while the *Separation Movement and Offer* trading mode had a drop in reward for $n = 2$, but increased overall afterwards. Therefore we can conclude that the trading mode *Separation Movement and Offer* does achieve higher rewards for higher amounts n of offer actions. Furthermore the agents in the *Original Action Trading* completed more trades than with *Separation Movement and Offer*. Especially at $n = 1$ the agents traded 2.8 times on average, while they only traded 1.79 times with *Separation Movement and Offer*. This difference is rather



(a) Original Action Trading Reward

(b) Separation Movement and Offer Reward

(c) Original Action Trading Steps

(d) Separation Movement and Offer Steps

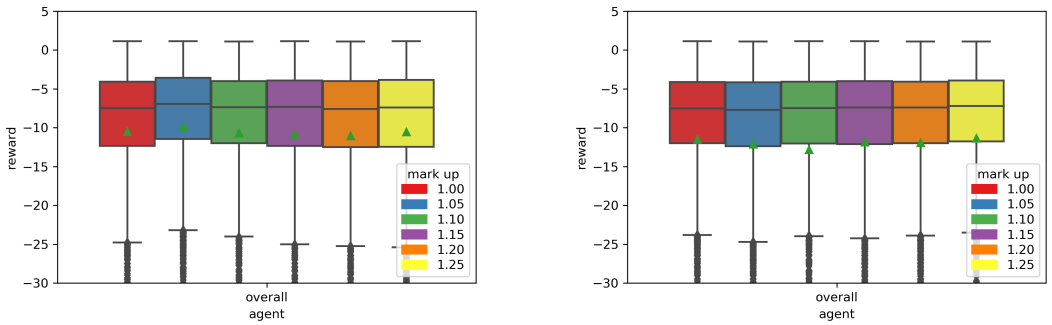
Figure 6.4: Shown are the training processes of 20 runs with 1500 episodes. The data in all figures are smoothed over 100 episodes. While agents with *Original Action Trading* mode (left) fail to learn for higher n, the learning curves of *Separation Movement and Offer* (right) are stable across $n \in [1, 5]$.

strange, as we should expect approximately the same amount of trades. In addition the increase of compensation 5.1 with M_c from 1.0 to 1.05 did not gain the same benefit for both trading modes. While in the *Original Action Trading* the increase of M_c also increased the reward, the reward decreased for agents with *Separation Movement and Offer*. Therefore we can only conclude that both action trading modes show significantly different behaviour. Nevertheless the *Separation Movement and Offer* mode enables the action trading to scale n and does not receive any penalty for doing so, while the *Original Action Trading* fails at higher n. The figure 6.4 shows the training process of agents with the different trading modes. There 6.4a the agents with the *Original Action Trading* mode are able to learn the approximated optimal policy π for $n \in [1, 3]$, but fail for $n = 4, 5$. This is also reflected in the average amount of steps for an episode, seen in 6.4c. The necessary steps decrease for $n \in [1, 3]$, where $n = 3$ already show very high amount of steps during the training process. For $n = 4, 5$ the agents need more and more steps to complete the episode, which indicates a rather poorly learning process. On the other hand agents with the trading mode *Separation Movement and Offer* show a constant good learning process across $n \in [1, 5]$, displayed in 6.4b. Also the steps the agents need to complete an episode decrease throughout the training process, which are shown in 6.4d. Overall agents do not gain benefit from long-term trades and more offer actions n.

6.3 Parameters

Cooperation depends on many factors. In this section we want to investigate the impact of different parameters on the cooperation between agents. In particular we want to inspect the variation of mark up M_c and budget B and their consequences on the behaviour of cooperative agents with action trading.

While we already increase the mark up M_c of the compensation 5.1 by 5% in 6.2.1



(a) Mark up: Original Action Trading

(b) Mark up: Separation Movement and Offer

Figure 6.5: Mark up Comparison: Shown are the rewards for different mark ups of agents with action trading and one offer action $n = 1$. For trading mode *Original Action Trading* we see an significant increase of reward for $M_c = 1.05$. The increase of mark up for *Separation Movement and Offer* shows decreasing reward for $M_c = 1.05$, but an increase afterwards.

and 6.2.2, we want to raise it even further. We therefore trained agents with $n = 1$ offer actions in 20 runs for 1500 episode each and then created the evaluation data based on additional 20×2000 episodes. We repeated this process for the mark ups $M_c \in [1.0, 1.05, 1.1, 1.15, 1.2, 1.25]$. The results are shown in 6.5a, where the overall rewards of the different mark ups M_c are displayed. Based on these results we can assume, that the mark up $M_c = 1.05$ is the most attractive for agents with the trading mode *Original Action Trading*, as the agents have the highest reward with the mean of $R_{1.05} = -9.87$ and median of $R_{1.05} = -6.94$. For other mark ups the agents achieve less reward. Repeating the comparison of different mark ups for the trading mode *Separation Movement and Offer* we see an overall drop in the reward for $M_c = 1.05$ in figure 6.5b. For higher mark up the reward increases again up to the point were it exceeds $M_c = 1.0$. While the mean reward of agents with $M_c = 1.0$ is $R_{1.0} = -11,49$ and median is $R_{1.0} = -7.5$, the mark up of $M_c = 1.25$ shows better results with the mean of $R_{1.25} = -11.34$ and median of $R_{1.25} = -7.2$. Thus the agents need more mark up for trading mode *Separation Movement and Offer* to benefit from trades with $n = 1$ offer actions, whereas the agents with *Original Action Trading* perform better for $M_c = 1.05$.

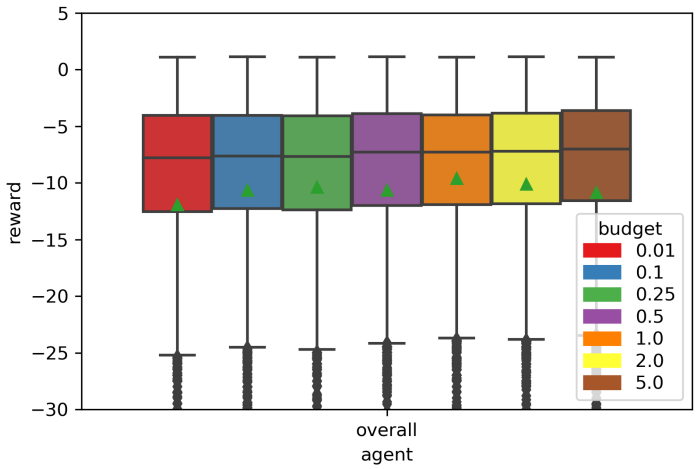


Figure 6.6: Shown are the rewards for different budgets B of agents with one offer action $n = 1$ and trading mode *Original Action Trading*. With increasing amount of budget the agents are able to perform better, which shows that budgeting does not benefit agents.

For the trading budget B we only compare the agents with the trading mode *Original Action Trading* and one offer action $n = 1$. We use the mark up of $M_c = 1.05$ as this amount of compensation showed the highest reward in the mark up comparison, also shown in figure 6.5a. Comparing the results from 20×2000 episodes after 20 runs with 1500 episodes in figure 6.6, we see an increase in the mean of the reward as we increase the trading budget $B \in [0.01, 1.0]$ and an overall increase in the median. The rewards constantly increase as the agents are able to trade more. While the median of the agents with the budget $B = 0.01$ is -7.78, it climbs up to -7.0 for $B = 5.0$. A comparison of higher budget is not necessary as the agents are only able to process 3 machines which result in a maximum of 3.0 reward. We therefore assume that a limited budget in this

specific setting is only harming the performance of the agents. Additionally there is no sweet spot for the budget, where the agents would surpass others.

6.4 Payment Timing

The timing of the compensation transfer is important for cooperation, as the offering agent loses reward before or after the offer actions are completed. Agents want to maximize their immediate and future reward; depending on the discount factor γ the importance of both vary 3.1. If the discount factor is less than 1, $\gamma < 1$, the agents care more about the immediate reward. Thus we expect the agents to trade less if the payment timing is before the offer execution, as they lose reward before seeing any gain of reward. On the other hand agents who pay afterward have a more positive mindset as they only see a gain in reward regardless of their care on the immediate or future reward. Another behaviour we can potentially investigate by shifting the payment timing is the conditioning of agents, as suspected in the action trading [SBGP18]. To verify our assumptions and examine the conditioning we dropped the trade related channels of the observation in this experiment and used the trading mode *Original Action Trading* with one offer action $n = 1$ and mark up $M_c = 1.05$. By training the agents in 20 separate runs with each consisting of $50,000$ episodes a behaviour pattern emerged. The results are shown in figure 6.7, where we see the training process of the agents with different payment timings. From the results we can observe the decreasing trend of trade completions as the episodes advance. This decrease applies to both payment timings, after and before. We assume the trend for the after payment timing emerges from the conditioning which was already proposed in [SBGP18]. The following agent assumes that there is an offer and tried to complete the

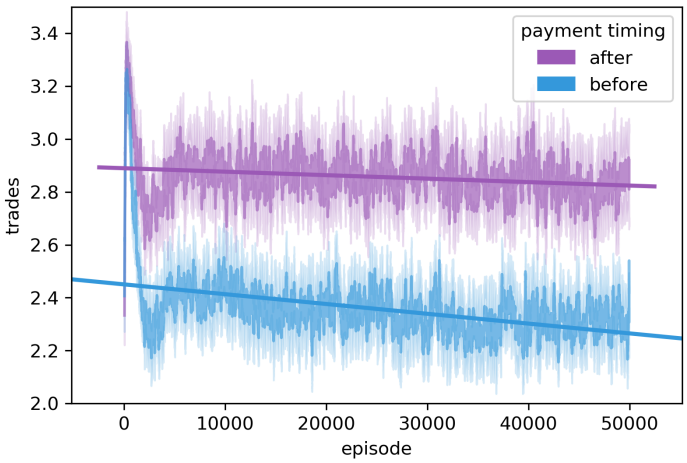


Figure 6.7: Shown are the amount of trades completed during the training process of $20 \times 50,000$ episodes for different payment timings. While agents with payment afterward complete on average 2.86 trades, agents with payment beforehand only complete 2.36. The regression lines shown for both payment timings are: after: $-1.21e^{-6}x + 2.89$ and before: $-3.69e^{-6}x + 2.45$.

offer actions, but there was no offer in the first place. By not providing any offer the offering agent does not have to pay any compensation and therefore only condition the other agent. While the decrease of trades for the payment timing afterward is $-1.21e^{-6}$ per episode, the amount of trades for the before payment drops with $-3.69e^{-6}$. This more than three times faster decrease for the before payment timing may due to the previous proposed maximization of reward. Agents want to maximize their immediate over their future reward, as we use the discount factor of $\gamma = 0.95$. This shortsighted reward maximization intimidates agents to make offers, as they lose reward immediately. This behaviour can be observed in the figure 6.7 as the agents who pay afterwards complete 2.86 trades on average, whereas agents who pay beforehand only trade 2.36 times. This intimidation also leads to the way higher decrease over the episodes in the before payment timing. If we exclude the first $10,000$ episodes of the results where agents still have to figure out the best trade amount, we see the difference in decrease between both timings weakened. Thus the trades for the after payment decrease with $-1.42e^{-6}$ per episode, whereas they decrease with $-2.10e^{-6}$ for the before payment. Nevertheless we can say they drop faster for the payment beforehand. We conclude that there is conditioning of agents for the after payment. Additionally we see the before payment timing with $\gamma < 1$ decrease the willingness of agents to cooperate tremendously. But we can not make any assumption regarding the conditioning for the before payment timing, as $\gamma < 1$ does not make the payment timings indifferent from one another. Therefore further investigations with $\gamma = 1$ are necessary.

6.5 Partial Payment

Following a series of actions can be difficult for agents as the environment constantly changes and therefore create new opportunities. Since they want to take the most promising actions in their possession, they may dismiss the trade. This creates a loss in reward as the agent followed the offer only for some offer actions and therefore did not utilize the full potential of gaining reward. Due to the incomplete offer the following agent do not receive any reward. To compensate this loss we introduced partial payment, where agents get compensated even if they only follow $i < n$ offer actions. To investigate the emerging cooperative behaviour of this approach, we trained agents in 20 runs for 1500 episodes where they can offer a trade consisting of 15 offer actions. After the training process we let the agents complete additional 2000 episodes, which our evaluation is based on. We repeated this procedure for $M_c \in [1.0, 1.05, 1.1, 1.15, 1.2, 1.25]$. The amount of trades the agents complete during 20×2000 episodes for the trading mode *Separation Movement and Offer* are shown in 6.8. There the agents overall completed for the most part one offer action $i = 1$ with the amount of 0.5658 and exponential decreasing amount of following for $i \in [1, 14]$. With increasing amount of offer actions, the followed average drops until $i = 14$ with 0.0007 trades. For $i = n = 15$ the amount raises again with 0.0027 trades, this may due to the limit of $n = 15$ offer actions agents can offer. According to 6.8 we can conclude that agents rather cooperate short-term, as the amount of offer actions followed are higher for lower i. On the other hand agents do not want to cooperate long-term, as the curves decrease exponentially with i. Besides the cooperation trend we can perceive that agents willingness to trade is higher for certain mark ups $M_c \in [1.0, 1.05, 1.15, 1.2]$.

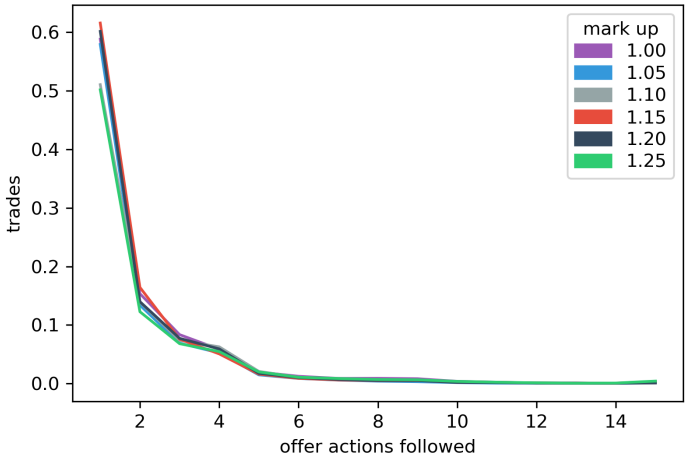


Figure 6.8: Partial Payment (*Separation Movement and Offer* Mode): For executing i actions the agent get paid even for not following the complete offer with n actions. Shown are the average amount of offer actions followed per episode by both agents during 20×2000 episodes after 20 runs of 1500 training episodes. Agents rather trade short- than long-term.

7 Conclusion

In conclusion the adopted social approach of action trading also works for the smart factory domain. While we modified the original action trading of [SBGP18] to fit the environment, we also proposed a new action trading mode. All the results in 6 should be taken with a grain of salt, as the experiments showed high fluctuations. However, we used 20 different runs for every experiment to make our evaluation more representative. For both trading modes agents with action trading were able to outperform the non cooperative agents. While the action trading performs well for $n = 1$ offer actions, scaling the concept to $n \in \mathbb{N}$ show mixed results. The trading mode *Original Action Trading*, which is a direct adaptation of the action trading in [SBGP18], performs poorly for $n \geq 2$. This is caused by the exponential growth of the action space with n. To solve this problem we introduced a new trading mode called *Separation Movement and Offer*, where we keep the action space constant. This trading mode showed a more stable learning process compared to the *Original Action Trading*. Furthermore the new trading mode showed higher impact on the reward and enabled us to compare the results of different n. There we observed a drop in reward until $n = 2$, but a recovering trend for higher n. During the scaling of the offer actions n we examined the cooperation between agents. While agents completed more trades for lower n, the amount drops significantly as we increase n. This gives us insight into the social behaviour of the agents regarding the duration of cooperation. The agents prefer to cooperate short-term rather than long-term. The experiment of partial payment in 6.8 shows this trend directly, where the agents were given the possibility to follow an offer as long as they wish. A factor which influences the willingness to cooperate longer is the mark up M_c. While we only see a minor change in trading behaviour, we see a high impact on the reward. In particular the mark up $M_c = 1.05$ show a major increase in reward for agents with the trading mode *Original Action Trading*. Nevertheless further investigation is needed, as we only compared different mark ups in high intervals of 5%. A more dense analysis could provide more information on the optimal mark up for agents in this environment. The trading budget shows a minor impact on the reward as well. The more limited the budget is, the worse the agents perform. Since we only increase the budget in large steps, we potentially missed out on a benefit from a limited budget. An major factor which influences the cooperation between agents is the payment timing. The results show way higher willingness to trade for agents with the payment timing being afterwards rather than beforehand. Moreover, conditioning should not be neglected, as the amount of trades decreases in the course of episodes. Further research on conditioning is therefore necessary. The social behaviour of the agents are also influenced by the trading modes. Agents with the *Original Action Trading* completed noticeable more trades than agents with *Separation Movement and Offer*. In the evaluation 6.2.2 we assumed this may due to the batch size and the expanding memory entries, but could also be caused by a different factor. More research is needed to make a more meaningful proposal.

List of Figures

Bibliography

[BEH+18] Yoram Bachrach, Richard Everett, Edward Hughes, Angeliki Lazaridou, Joel Z. Leibo, Marc Lanctot, Mike Johanson, Wojtek Czarnecki, and Thore Graepel. Negotiating team formation using deep reinforcement learning. 2018.

[CEW11] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. volume 5, 10 2011.

[HKT18] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. Is multiagent deep reinforcement learning the answer or the question? A brief survey. *CoRR*, abs/1810.05587, 2018.

[Kra93] Sarit Kraus. Agents contracting tasks in non-collaborative environments. In *AAAI*, 1993.

[Lit94] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

[LP17] Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *CoRR*, abs/1707.01068, 2017.

[LPB16] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *CoRR*, abs/1612.07182, 2016.

[LZL+17] Joel Z. Leibo, Vinícius Flores Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *CoRR*, abs/1702.03037, 2017.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 02 2015.

[SB98] R.S. Sutton and A.G. Barto. *Reinforcement learning: an introduction*. MIT Press, Cambridge, MA., 1998.

[SBGP18] Kyrill Schmid, Lenz Belzner, Thomas Gabor, and Thomy Phan. Action markets in deep multi-agent reinforcement learning. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning (ICANN 2018)*, pages 240–249, Cham, 2018. Springer International Publishing.