

Design of artificial intelligence agent for supply chain manage- ment using deep reinforcement learning based on NegMAS Library

MASTERARBEIT

KIT – KARLSRUHER INSTITUT FÜR TECHNOLOGIE
FRAUNHOFER IOSB – FRAUNHOFER-INSTITUT FÜR OPTRONIK,
SYSTEMTECHNIK UND BILDAUSWERTUNG

Ning Yue

10. Mai 2021

Verantwortlicher Betreuer:	Prof. Dr.-Ing. habil. Jürgen Beyerer
Betreuende Mitarbeiter:	Dr.-Ing. Tim Zander
	Prof. Dr.-Ing. Yasser Mohammad(Extern)

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung beachtet habe.

Karlsruhe, den 10. Mai 2021

(Ning Yue)

Abstract

Consider an agent that can cooperate with others and autonomously negotiate, to reach an agreement. These agents could achieve great score, e.g. profitability of factory in realistic. Such agent is practice in complex, realistic environment such as supply chain management. In experiments come from this paper, learnable agents are proposed with Multi-Agent deep reinforcement learning method(QMIX and MADDPG) to achieve the goal. The strategy of agent will be improved when continuously interacting with others, central training but executing with local observation. The learned strategy enable autonomous agents to negotiate concurrently with multiple different type, unknow opponents in real-time, over complex multi-issues.

Kurzfassung

Falls die Abschlussarbeit auf Deutsch geschrieben wird, genügt die deutsche Kurzfassung.

Notation

Lower case letters are used for the values of random variables and for scalar functions. Capital letters are used for random variables and major algorithms variables.

General identifier

α, \dots, ω	Skalare
a, \dots, z	Skalar, Vektor, Funktionssymbol (oder Realisierung einer Zufallsvariablen)
$\mathbf{a}, \dots, \mathbf{z}$	Zufallsvariable (skalar bzw. vektoriell)
$\hat{\mathbf{a}}, \dots, \hat{\mathbf{z}}$	Schätzer für jeweilige Variable als Zufallsgröße
\hat{a}, \dots, \hat{z}	Realisierter Schätzer für jeweilige Variable
A, \dots, Z	Matrix
$\mathbf{A}, \dots, \mathbf{Z}$	Matrix als Zufallsgröße
$\mathcal{A}, \dots, \mathcal{Z}$	Menge
$\mathfrak{A}, \dots, \mathfrak{Z}$	Mengensystem

Special identifier

s	State
a	action
\mathcal{S}	set of nonterminal states
$\mathcal{A}(s)$	set of actions possible in state s
\mathcal{R}	set of possible rewards
t	discrete time step
T	final time step of an episode
S_t	state at t
A_t	action at t
R_t	reward at t
G_t	return (cumulative discounted reward) following t
π	policy, decision-making rule
$\pi(s)$	action taken in state s under deterministic policy π
$\pi(a s)$	probability of taking action a in state s under stochastic policy π
$p(s', r s, a)$	probability of transitioning to state s' , with reward r , from s , a
$v_\pi(s)$	value of state s under policy π (except return)
$v_*(s)$	value of state s under the optimal policy

General quantities

\mathbb{C}	Menge der komplexen Zahlen
\mathbb{H}	Poincaré Halbebene
\mathbb{N}	Menge der natürlichen Zahlen (ohne Null)
\mathbb{N}_0	Menge der natürlichen Zahlen mit Null
\mathbb{Q}	Menge der rationalen Zahlen
$\mathbb{Q}^{>0}, \mathbb{Q}^{<0}$	Menge der positiven bzw. negativen rationalen Zahlen
\mathbb{R}	Menge der reellen Zahlen
$\mathbb{R}^{>0}, \mathbb{R}^{<0}$	Menge der positiven bzw. negativen reellen Zahlen
\mathbb{Z}	Menge der ganzen Zahlen

Special symbols

$\mathfrak{N}(\mu, \sigma^2)$	Normalverteilung mit Erwartungswert μ und Varianz σ
$\mathfrak{F}_{r,s}$	Fisher-Verteilung mit r Zähler- und s Nennerfreiheitsgraden
t_s	Student- t -Verteilung mit s Freiheitsgraden
δ_ξ	Ein-Punkt-Maß an der Stelle ξ
χ_s^2	χ^2 -Verteilung mit s Freiheitsgraden

Contents

1	Introduction	1
1.0.1	Motivation	1
1.0.2	Outline of this Work	2
2	Background	5
2.1	Game theory	5
2.1.1	Nash Equilibrium	5
2.1.2	Pareto Efficient	5
2.1.3	Markov Games	5
2.2	Autonomous Negotiaion	6
2.2.1	Utility Function	6
2.2.2	Rubinstein bargaining mechanism	7
2.2.3	Stacked alternating offers mechanism(SAOM)	7
2.3	Artificial Intelligence	7
2.3.1	Sub-areas	7
2.3.2	Methods	8
2.3.3	Application Field	9
2.4	Artificial Neural Network	9
2.5	Reinforcement Learning	9
2.5.1	The Agent–Environment Interface	9
2.5.2	Value Function	11
2.5.3	Bellman Functions	11
2.5.4	Q-Learning	11
2.5.5	Policy Gradient PG	12
2.5.6	Deep Reinforcement Learning (DRL)	12
2.6	Platform and Library	13
2.6.1	GENIUS	13
2.6.2	NegMAS	13
2.6.3	SCML	13

2.6.4	PyTorch	14
2.6.5	OpenAI Gym	14
2.6.6	Ray	16
3	Related Works	17
3.1	Heuristic Negotiation Strategies for Autonomous Negotiation	17
3.1.1	Time-based Strategy (Aspiration Negotiator)	17
3.1.2	Concurrent Negotiation Strategy (CNS)	17
3.1.3	Conclusion	17
3.2	Reinforcement Learning used in Autonomous Negotiation	18
3.3	Challenges in Deep Reinforcement Learning	18
3.3.1	Sparse Reward	18
3.3.2	Non-stationary environment	18
3.3.3	Huge action space	19
4	Analyze	21
4.1	NegMAS with OpenAI Gym	21
4.1.1	Configuration	21
4.1.2	Model	22
4.1.3	Single-Agent Environment	22
4.1.4	Game	23
4.1.5	Challenges of the environment	23
4.1.6	Analysis of the reinforcement learning algorithms	24
4.1.7	Conclusion	24
4.2	SCML with OpenAI Gym	24
4.2.1	Configuration	24
4.2.2	Model	25
4.2.3	Multi-Agent Environment	25
4.2.4	Scenario	26
4.2.5	Challenges of the environment	27
4.2.6	Analysis of the reinforcement learning algorithms	27
4.2.7	Conclusion	28
5	Methods and Experiments	29
5.1	Single-Agent Bilateral Negotiation Environment (SBE)	29
5.1.1	Independent Negotiator in NegMAS	29

5.1.2	Experiment	30
5.1.3	Evaluation	31
5.2	Multi-Agent Concurrent Bilateral Negotiation Environment (MCBE)	31
5.2.1	MADDPG in SCML	31
5.2.2	QMIX in SCML	32
5.2.3	Experiment	33
5.3	Conclusion	34
6	Conclusions and Future Work	37
6.1	Others goal	37
6.2	Evaluation	37
6.3	Design of reward function	37
6.4	Complex environment	37
6.5	Huge scale high performance learning	37
	Bibliography	39
	List of Tables	41
	List of Figures	43
	List of Theorems	45
	Listings	47
	Glossary	49

1 Introduction

Computer software and hardware development leads to the appearance of non-human software agencies. An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [1].

A supply chain is a network of suppliers, factories, warehouses, distribution centers and retailers, through which raw materials are acquired, transformed, produced and delivered to the Customer[2]. In the network we could find many entities whose could be considered as agents in the Multi-agent systems (MAS). Multi-agent System(MAS) are suitable the domains that involve interactions between different people or organizations with different (possibly conflicting) goals and proprietary information, there are many approaches are proposed in order to solve the problem in the supply chain mangement system. Such as negotiation-based Multi-agent System[3].

The Supply Chain Management (SCM) world designed in SCML by Yasser Mohammad simulates a supply chain consisting of multiple factories that buy and sell products from one another. The factories are represented by autonomous agents that act as factory managers. Agents are given some target quantity to either buy or sell and they negotiate with other agents to secure the needed supplies or sales. Their goal is to turn a profit, and the agent with the highest profit (averaged over multiple simulations) wins [Moh+19]. It is characterized by profit-maximizing agents that inhabit a complex, dynamic, negotiation environment[4]. There are two games built on the top of NegMAS which is the library for developing autonomous negotiation agents embedded in simulation environments.

In economic, autonomous agent can be considered as a specific type of agent, with a focus on generating economic value. This technology will be at the forefront of the next industrial revolution, affecting numerous billion dollar industries such as transportation and mobility, finance, supply chain, energy trading, social networks and Marketplaces and e-commerce.

1.0.1 Motivation

Negotiation is a complex problem, in which the variety of settings and opponents that may be encountered prohibits the use of a single predefined negotiation strategy. Hence the agent

should be able to learn such a strategy autonomously []. By autonomy it means “independent or self-governing”. In the context of an agent, this means it can act without constant interference from its owner []. Autonomous negotiation agent is meaningful in many realistic environment, such as all mentioned economic value in industries. The development of current machine learning algorithms and increased hardware resource make it possible, model the realistic environment to evaluate the problem with computer system. According to the modeled realistic environment, it will be easier to find more possible solutions with the help of machine learning technology.

In this work, we use some modeled negotiation environments, such as single agent environment (bilateral negotiation), and analyze whether deep reinforcement learning can be used to let agent learns some strategies autonomously in these environments. In contrast to single agent environment, in the supply chain environment, there are many agents with the same goal. After analyzing the simple environment, we need to explore whether multi-agent deep reinforcement learning can be used to obtain better results in multi-agent environment.

How good strategy can be learned by deep reinforcement learning in single agent environment(bilateral negotiation)?

How good strategy can be learned by multi-agent deep reinforcement learning in multi-agent environment(concurrent negotiation)?

What is the difference between deep reinforcement learning strategies and other heuristic strategies?

1.0.2 Outline of this Work

In the following, the other chapters of this work are listed and their content briefly presented.

Chapter 2: Background: This chapter contains basic knowledge and concepts that are necessary to understand the thesis. Firstly, some concepts from game theory are listed. These concepts are often discussed and used in autonomous negotiation. Secondly, utility function, some negotiation mechanisms are described in the section on autonomous negotiation. In addition, the basics and the historical development of artificial intelligence are presented. The focus of this chapter is on reinforcement learning.

Chapter 3: Related Works In this chapter, some published matter which technically relates to the proposed work in this thesis will be discussed. These publication will be divided as three categories: Negotiation Strategies for Autonomous Negotiation, Reinforcement Learning used in Autonomous Negotiations and Challenges in Deep Reinforcement Learning. In the section

Challenges in Deep Reinforcement Learning, some related algorithms except RL. will be discussed.

Chapter 4: Analyze

Chapter 5: Methods and Experiments

Chapter 6: Conclusions and Future Work

2 Background

2.1 Game theory

2.1.1 Nash Equilibrium

The concept of a Nash equilibrium plays a central role in noncooperative game theory[]. The definition in simple setting of a finite player is described as follow. I players indexed by $i=1,...,I$. The strategy of agent i is s_i choosed from N_i pure strategies. A strategy profile of all agents written as $s = (s_1, s_2, s_I)$, $s_i|s_i'$ for the strategy profile $(s_1, \dots, s_{i-1}, s_{i+1}, s_I)$, or the s with the part of i changed from s_i to s_i' . For each player i and strategy s , $u_i(s)$ denotes i 's expected utility[].

Proposition 2.1 (Nash Equilibrium) *For each agent i and s_i' , $u_i(s) \geq u_i(s|s_i')$*

In terms of words description, the definition of Nash equilibrium is that if other agents do not change its strategy, then no single agent can obtain higher utility.

The learning process of RL-Agent in this paper can be considered as an incomplete information static non-cooperative game.

2.1.2 Pareto Efficient

Proposition 2.2 (Pareto Efficient) *no other feasible allocation $\{x'_1, \dots, x'_n\}$ where, for utility function u_i for each agent i , $u_i(x_i)$ for all $i \in \{1, \dots, n\}$ with $u_i(x'_i) > u_i(x_i)$ for some i [].*

Pareto Efficient is a state at which resources in a system are optimized in a way that one dimension cannot improve without a second worsening.

2.1.3 Markov Games

Finite Markov Decision Processes

Multi-Agent Markov Decision Processes Methods mentioned in the paper are based on a multi-agent extension of Markov decision processes(MDPs) called partially observable Markov games. There are N players indexed by $n = 1, 2, \dots, N$.

2.2 Autonomous Negotiation

Negotiation is an important process in coordinating behavior and represents a principal topic in the field of multi-agent system research. There has been extensive research in the area of automated negotiating agents.

Automated agents can be used side-by-side with a human negotiator embarking on an important negotiation task. They can alleviate some of the effort required of people during negotiations and also assist people who are less qualified in the negotiation process. There may even be situations in which automated negotiators can replace the human negotiators. Thus, success in developing an automated agent with negotiation capabilities has great advantages and implications[].

Through the negotiation agents, many problems that arise in real or simulated domain can be solved. In industrial domains, In commercial domains, the Supply Chain Management System (SCMS) functionality is implemented through agent-based negotiation environment, in which contracts can be signed through negotiation between agents. Many papers describe ongoing effort in developing a Multi-agent System (MAS) for supply chain management[SCML].

In game domains, bilateral negotiation in [GENIUS]

2.2.1 Utility Function

Utility function is an important concept in economics. It measures preferences over a set of goods and services. Utility represents the satisfaction that consumers receive for choosing and consuming a product or service[]. In NegMAS and SCML, utility function could measure either single offer or set of offers.

Utility is measured in units called utils, but calculating the benefit or satisfaction that consumers receive from is abstract and difficult to pinpoint[]. In the package NegMAS, SCML are built-in some utility functions, through inheritance of these it is easy to design new Utility function by developer. Such as linear utility function and real utility function OneShotUfun designed in SCMLOneShotWorld.

[1] linear utility function [2] real utility function OneShotUfun

It is an important point for designing a new Agent in autonomous negotiation environments. For heuristic agents utility function is a keypoint to measure preferences. For reinforcement learning agents utility function conducts the behavior of learnable agents, used as a part of reward function, significantly affect the design and evaluation of RL-Agent.

2.2.2 Rubinstein bargaining mechanism

Rubinstein bargaining mechanism is widely cited for multi-round bilateral negotiation [Rub82]. Two agents in the mechanism which has an infinite time horizon have to reach an agreement. To the state of nash equilibrium.

2.2.3 Stacked alternating offers mechanism(SAOM)

SAOM is also named as stacked alternating offers protocol. Agents can only take their action when it is their turn. SAOM allows negotiating agents to evaluate only the most recent offer in their turn and accordingly they can either accept offer, make a counter offer or walk away.

In the SCML OneShotWord, at the first step all of the agents will propose a first offer.

2.3 Artificial Intelligence

Artificial Intelligence is a broad branch of computer science that is focused on a machine's capability to produce rational behavior from external inputs. The goal of AI is to create systems that can perform tasks that would otherwise require human intelligence[.].

There is a set of three related items that sometimes are erroneously used interchangeably, namely artificial intelligence, machine learning, and neural networks. According to Encyclopaedia Britannica, AI defines the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. On the other hand, according to H.A. Simon, one of the pioneers of the field, machine learning is a "field of study that gives computers the ability to learn without being explicitly programmed"

2.3.1 Sub-areas

Fig. 2.1 shows the relationship of artificial intelligence, machine learning and deep learning.

Artificial Intelligence

Artificial intelligence, also called machine intelligence, can be understood by an intelligence, unlike the natural intelligence shown by humans and animals, which is demonstrated by machines. It looks at ways of designing intelligent devices and systems that can address problems creatively that are often treated as a human prerogative. Thus, AI means that a machine somehow imitates human behavior.

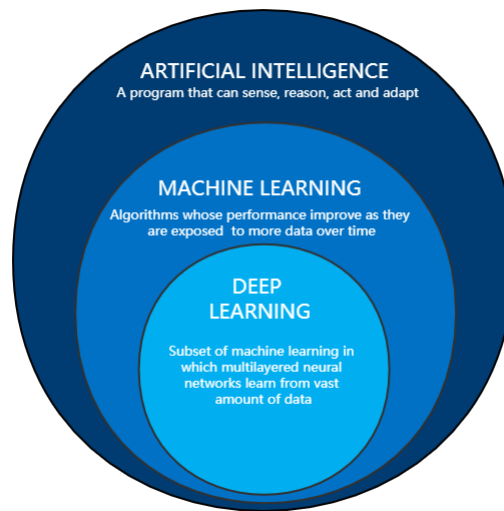


Figure 2.1: Sub-areas of artificial intelligence [SUM20]

Machine Learning

Machine learning is an AI subset and consists of techniques that enable computers to recognize data and supply AI applications. Different algorithms (e.g., neural networks) contribute to problem resolution in ML.

Deep Learning

Deep learning, often called deep neural learning or deep neural network, is a subset of machine learning that uses neural networks to evaluate various factors with a similar framework to a human neural system. It has networks that can learn from unstructured or unlabeled data without supervision.

2.3.2 Methods

Supervised Learning Training data contains optimal outcomes (also known as inductive learning). Learning is tracked in this method. Some famous examples of supervised machine learning algorithms are Linear regression for regression problems.

Unsupervised Learning There are not the desired outputs in the training results. Clustering is an example. It is impossible to know what is and what is not good learning.

Semi-supervised Learning A few desired outputs are included in the training data.

Reinforcement Learning Rewards are given after a sequence of actions. In a given case, it is a matter of taking appropriate steps to maximize compensation. It is the most ambitious method of learning in AI.

2.3.3 Application Field

eCommerce

Logistics and Supply Chain

Artificial intelligence (AI) researchers have paid a great deal of attention to automated negotiation over the past decade and a number of prominent models have been proposed in the literature.

Tools for computer science

2.4 Artificial Neural Network

2.5 Reinforcement Learning

2.5.1 The Agent–Environment Interface

The reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. These interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent [SB18]. Figure 2.2 diagrams the agent–environment interaction.

Above process is a typical single-agent interaction process. Single agent reinforcement learning algorithms are based on this process. By extending this interactive process, multi-agent interactive process can be intuitively diagrammed In 2.3. These learning cases are called Multi-Agent-Reinforcement Learning MARL. In the figure 2.3 agent is splited as two parts perceptor and learner. Perceptors observe the environment and send the state to learners. Learners learn strategies based on the states, rewards and actions. There are many methods, which proposed in these years. Some MADRL methods will be discussed in following sections. According to the design requirements of the training environment, the structure of the interaction process is flexible.

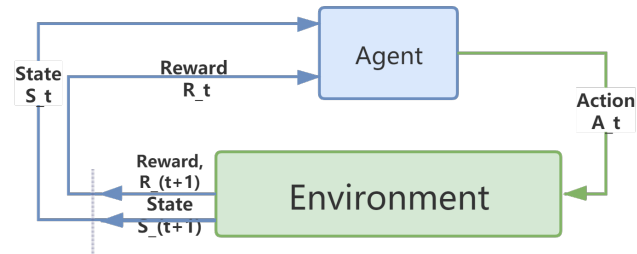


Figure 2.2: The agent–environment interaction in reinforcement learning, diagrammed In [SB18].

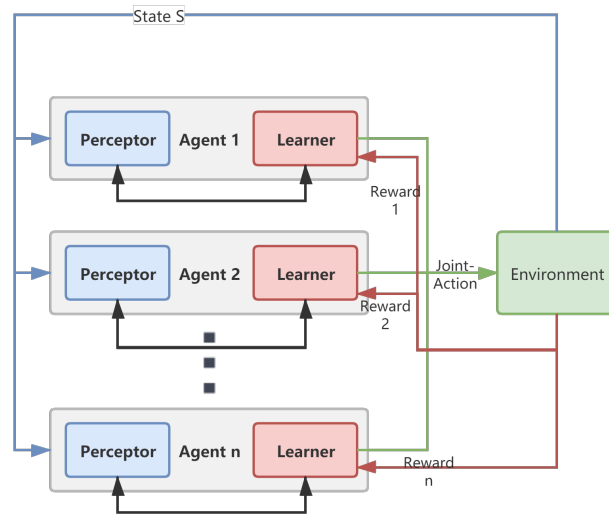


Figure 2.3: The multi-agent–environment interaction in reinforcement learning, diagrammed In [Fan+20].

2.5.2 Value Function

Value Function can be referred as the state-value function and state-action-value-function which consists fixed state or both state and action, respectively.

State-Value-Function measures the goodness of each state. It is based on the return Reward G following a policy π . In a formal way, the value of $V_\pi(s)$ is:

$$V_\pi(s) = \mathbb{E}_\pi [G_t \mid s = s_t] = \mathbb{E}_\pi \left[\sum_{j=0}^T \gamma^j r_{t+j+1} \mid s = s_t \right] \quad (2.1)$$

Action-Value Function which measures the goodness of each pair of state, action. Compared with state-value function action is determined.

$$Q_\pi(s, a) = \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{j=0}^T \gamma^j r_{t+j+1} \mid S_t = s, A_t = a \right] \quad (2.2)$$

2.5.3 Bellman Functions

In summary, Bellman Functions decomposes the value function into two parts, the immediate reward plus the discounted future values. Equation 2.3 show how to recursively the Bellman equation is defined for the state-value function:

$$V_\pi(s) = \sum_a \pi(a \mid s) \cdot \sum_{s'} P_{ss'}^a (r(s, a) + \gamma V_\pi(s')) \quad (2.3)$$

As same as Bellman equation for the state-value function, equation ?? tells us how to find recursively the value of a state-action pair following a policy π .

$$Q_\pi(s, a) = \sum_{s'} P_{ss'}^a (r(s, a) + \gamma V_\pi(s')) \quad (2.4)$$

2.5.4 Q-Learning

To maximize the total cumulative reward in the long sequence is the goal of Agent. The policy, which maximize the total cumulative reward is called optimal policy formed as π^* . Optimal State-Action-Value-Function and optimal State-Value-Function are formed as $Q_{\pi^*}(s, a)$ and $V_{\pi^*}(s)$, respectively.

$$\mathcal{L}(\theta) = \sum_{i=1}^b [(y_i - Q(s, a \mid \theta))^2] \quad (2.5)$$

2.5.5 Policy Gradient PG

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s \sim p^{\pi}, a \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a | s) Q^{\pi}(s, a)] \quad (2.6)$$

2.5.6 Deep Reinforcement Learning (DRL)

Deep Q-Networks

Proximal Policy Optimization (PPO)

Disadvantage: the distribution of action changed too quickly, when the reward is always positive or negative, some possible action will disappear. PPO uses some constraint tricks to avoid it, such as clip of policy. The loss function based on PPO-clip is as follows.

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\pi_{\theta_k}}(s, a), \quad \text{clip} \left(\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right) \quad (2.7)$$

DPG, DDPG, MADDPG

DPG: create a μ function to determine the action instead of the sample in PG.

DDPG: use Actor-Critic model to create target and execute network. a variant of DPG. policy μ and critic Q_{μ} are approximated with deep neural networks.

MADDPG: DDPG used in multi-agents environments.

MADPG

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim p^{\mu}, a_i \sim \pi_i} [\nabla_{\theta_i} \log \pi_i(a_i | o_i) Q_i^{\pi}(\mathbf{x}, a_1, \dots, a_N)] \quad (2.8)$$

Extension to deterministic policies

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}} \left[\nabla_{\theta_i} \mu_i(a_i | o_i) \nabla_{a_i} Q_i^{\mu}(\mathbf{x}, a_1, \dots, a_N) \Big|_{a_i = \mu_i(o_i)} \right] \quad (2.9)$$

Loss function is

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'} \left[(Q_i^{\mu}(\mathbf{x}, a_1, \dots, a_N) - y)^2 \right], \quad y = r_i + \gamma Q_i^{\mu'}(\mathbf{x}', a'_1, \dots, a'_N) \Big|_{a'_j = \mu'_j(o_j)} \quad (2.10)$$

Value Decomposition Networks

Mixing Network used in QMIX

QMIX

2.6 Platform and Library

2.6.1 GENIUS

GENIUS: An integrated environment for supporting the design of generic automated negotiators [Lin+14].

2.6.2 NegMAS

NegMAS can model situated simultaneous negotiations such as SCM. Nevertheless, it can model simpler bilateral and multi-lateral negotiations.

NegMAS is a python library for developing autonomous negotiation agents embedded in simulation environments. The name negmas stands for either NEGotiation MultiAgent System or NEGotiations Managed by Agent Simulations. The main goal of NegMAS is to advance the state of the art in situated simultaneous negotiations. Nevertheless, it can; and is being used; for modeling simpler bilateral and multi-lateral negotiations, preference elicitation , etc.

NegMAS and Bilateral Negotiation NegMAS implements natively SAOM which can be set as bilateral negotiation mechanism.

2.6.3 SCML

A supply chain is a sequence of processes by which raw materials are converted into finished goods. A supply chain is usually managed by multiple independent entities, whose coordination is called **supply chain management(SCM)**. SCM exemplifies situated negotiation. The SCM world was built on top of an opensource automated negotiation platform called NegMAS to serve as a common benchmark environment for the study of situated negotiation [Moh+19]. This repository is the official platform for running ANAC Supply Chain Management Leagues. It will contain a package called scmlXXXX for the competition run in year XXXX. For example scml2019 will contain all files related to the 2019's version of the competition. There are three main different versions of SCML, which have different designs.

- SCML2020-OneShot

- SCML2020/2021
- SCML2019

SCML and Concurrent Bilateral Negotiations

SCML was originally developed as a part of NegMAS, from the version ? it was splited as an independent project to research SCM. SCML realized a SCM World to simulate the SCM process.

There are many agents which has same type in the SCM World.

Researchers have also developed many negotiation agents such as Agent1[], Agent2[], Agent3[] in GENIUS, Agent4[], Agent5[], Agent6[] in NegMAS.

2.6.4 PyTorch

PyTorch is an open source machine learning library and framework which performs immediate execution of dynamic tensor computations with automatic differentiation and GPU acceleration, and does so while maintaining performance comparable to the fastest current libraries for deep learning. [Pas+19]. While considering performance, it is also easier to apply and debug.

2.6.5 OpenAI Gym

OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms [Bro+16].

Environment

The core gym interface is Env, which is the unified environment interface. The following are the Env methods that developers should implement [Bro+16].

STEP Run one timestep of the environment's dynamics. When end of episode is reached, `reset()` is called to reset this environment's state. Accepts an action and returns a tuple (observation, reward, done, info).

- observation (object): agent's observation of the current environment, such as frame of the game.
- reward (float): amount of reward returned after previous action, such as 1 when action is go to left.

- **done** (bool): whether the episode has ended, in which case further `step()` calls will return undefined results, such as agent is dead in game, as `True`.
- **info** (dict): contains auxiliary diagnostic information (helpful for debugging, and sometimes learning), such as goal of agent.

RESET Resets the environment to an initial state and returns an initial observation. This function should not reset the environment's random number generator(s). Random variables in the environment's state should be sampled independently between multiple calls to `reset()`. Each call of `reset()` should yield an environment suitable for a new episode, independent of previous episodes.

RENDER Define how to display the output of the environment. Multiple modes can be used:

- **human**: Render to the current display or terminal and return nothing. Usually for human consumption
- **rgb_array**: Return a `numpy.ndarray` with shape `(x, y, 3)`, representing RGB values for an x-by-y pixel image, suitable for turning into a video.
- **ansi**: Return a string (`str`) or `StringIO.StringIO` containing a terminal-style text representation. The text can include newlines and ANSI escape sequences (e.g. for colors).

CLOSE Override `close` in the subclass to perform any necessary cleanup. Environments will automatically `close()` themselves when garbage collected or when the program exits. Save data at the end of the program.

SEED Sets the seed for this env's random number generator(s). It is useful for reproducing the results.

```
1 ob0=env.reset() #sample environment state , return first observation
2 a0=agent.act(ob0) #agent chooses first action
3 ob1,rew0,done0,info0=env.step(a0) #environment returns observation ,
4 #reward, and boolean flag indicating if the episode is complete.
5 a1=agent.act(ob1)
6 ob2,rew1,done1,info1=env.step(a1)
7 ...
8 a99=agent.act(o99)
9 ob100,rew99,done99,info2=env.step(a99)
```

```
10 # done99 == True => terminal
```

Listing 2.1: Logic of OpenAI Gym Interaction

From Listing 2.1, user can get the logic of interaction in OpenAI Gym.

Stable Baselines

The stable baselines developed in the project stable-baselines [Hil+18]. All implemented algorithms with characteristic discrete/continuous actions are shown in 2.1.

Name	Box	Discrete
A2C	Yes	Yes
ACER	No	Yes
ACKTR	Yes	Yes
DDPG	Yes	No
DQN	No	Yes
HER	Yes	Yes
GAIL	Yes	Yes
PP01	Yes	Yes
PP02	Yes	Yes
SAC	Yes	No
TD3	Yes	No
TRPO	Yes	Yes

Table 2.1: stable baselines algorithms

2.6.6 Ray

Ray is packaged with the following libraries for accelerating machine learning workloads.

- Tune: Scalable Hyperparameter Tuning
- RLlib: Scalable Reinforcement Learning
- RaySGD: Distributed Training Wrappers
- Ray Serve: Scalable and Programmable Serving

3 Related Works

In this chapter, related work on the relevant topics of this work is presented and discussed. The topics include autonomous negotiation, multi-agent reinforcement learning. At the last section, the work of reinforcement learning used in autonomous negotiation is presented.

3.1 Heuristic Negotiation Strategies for Autonomous Negotiation

3.1.1 Time-based Strategy (Aspiration Negotiator)

Type of aspiration is bouldware.

3.1.2 Concurrent Negotiation Strategy (CNS)

In a concurrent negotiation environment, an agent will negotiate with many opponents at the same time(one-to-many). One issue is how to coordinate all these negotiations. The author of the paper [Wil+12] designed an intuitive model with two key parts, namely the Coordinator and Negotiation Thread.

Negotiation Threads:The strategy of each negotiation thread is an extension of a recently published, principled, adaptive bilateral negotiation agent. This agent was designed to be used in a similarly complex environment, but only for negotiations against a single opponent.

Coordinator:The role of the coordinator is to calculate the best time, t_i and utility value, u_i at that time, for each thread. To do so, it uses the probability distributions received from the individual threads, which predict future utilities offered by the opponents.

3.1.3 Conclusion

From the analysis of the heuristic negotiation strategy in a specific field, we can get some important parameters, such as time, offer by opponent, that need to be considered as the information used in the RL. algorithm.

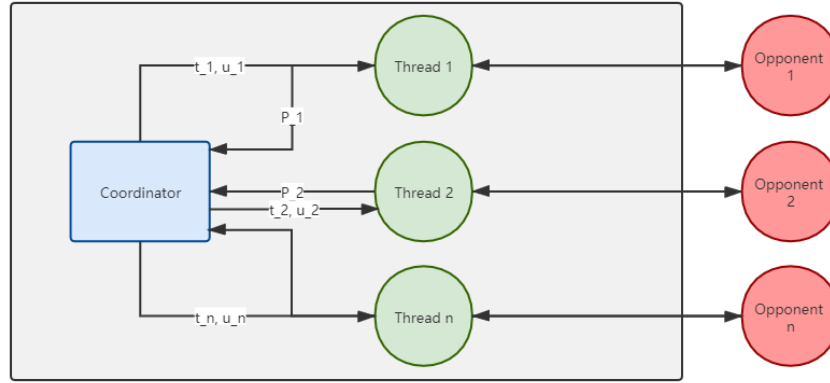


Figure 3.1: Architecture of the concurrent negotiation agent, best time: t_i and utility value: u_i , probability distributions: P [Wil+12].

3.2 Reinforcement Learning used in Autonomous Negotiation

NegoSI: A novel algorithm named negotiation-based MARL with sparse interactions (NegoSI) is presented by Luowei Zhou. In contrast to traditional sparse-interaction based MARL algorithms, NegoSI adopts the equilibrium concept and makes it possible for agents to select the non-strict Equilibrium Dominating Strategy Profile (non-strict EDSP) or Meta equilibrium for their joint actions [Zho+17].

RLBOA: From the paper [Bak+19] A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents.

ANEGMA: Work by [Bag+20]. A novel DRL-inspired agent model called ANEGMA, which allows the buyer to develop an adaptive strategy to effectively use against its opponents (which use fixed-but-unknown strategies) during concurrent negotiations in an environment with incomplete information.

3.3 Challenges in Deep Reinforcement Learning

3.3.1 Sparse Reward

Utility value as part of reward function reward shaping Curiosity Driven Imitation Learning

3.3.2 Non-stationary environment

The strategy of single agent is changed during training Multi-agent environment is non-stationary Multi-agent deep reinforcement learning

3.3.3 Huge action space

Action embedding, discrete action replaced by continuous action space.

4 Analyze

Two environments are developed for comparing the DRL algorithms used in this thesis: single-agent bilateral negotiation environment (SBE) and multi-agent concurrent bilateral negotiation environment (MCBE). The details are described in section 4.1.3 and 4.2.3. In addition to these environments, some methods have been implemented to make the training logic clearer, such as Game in section 4.1.4 and Scenario in section 4.2.4.

4.1 NegMAS with OpenAI Gym

NegMAS has implemented some negotiation mechanisms and specific simulated world, such as SAOM and SCML (Now as an independent project). In order to compare the algorithms in specific simulated world more easily, an interface is needed to connect NegMAS and RL algorithms. This interface and all algorithms can be rewritten from scratch, but it is very time-consuming and not ideal. The second option is to implement some RL framework interfaces, which will reduce a lot of work. OpenAI realize the environmental standardization and comparison of algorithms with the help of toolkit OpenAI Gym [Bro+16]. Although OpenAI Gym is not enough to complete the work in this thesis, the baseline algorithms and the environmental interface in the package greatly speed up the work. In this section, the implementation of environment and assisted methods used in bilateral negotiation will be presented.

With the help of OpenAI Gym, a bilateral negotiation environment can be developed on the top of SAOM to research reinforcement learning algorithms. OpenAI Gym implements many baseline algorithms, which can be easily tested in a custom environment.

4.1.1 Configuration

Negotiation Issues

NegMAS provides some classes and methods to design issues flexibly. In SBE following issues are used:

Price Integer between two values, such as (10, 20)

Quantity Integer between two values, such as (1, 10)

Time Relative step between zero and maximal step.

In the section Experiment 5.1.2 of Chapter Methods and Experiments 5, the configuration of the negotiation mechanism will be listed in detail.

4.1.2 Model

The model consists of five parts, environment SBE, negotiation game, negotiation mechanism, negotiator and reinforcement learning algorithms. Except for the negotiation mechanism mentioned and implemented in sections 2.2 and 2.6.2, others parts will be introduced step by step in following sections.

Entire model is shown in 4.1.

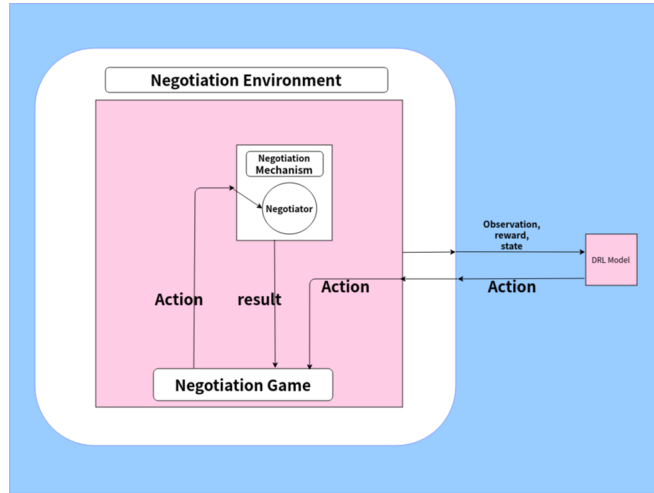


Figure 4.1: Model for single agent bilateral negotiation based on NegMAS

4.1.3 Single-Agent Environment

The interface of the OpenAI Gym Environments is designed for one agent as standard. Nevertheless, it must be examined how the SBE can be represented via this interface and controlled by the controller. The methods of the interface for the SBE are therefore defined below.

STEP Firstly, sets up but not performs the action received from RL. algorithms for the negotiator. Then, run the negotiation mechanism (such as SAOM) for one step. All actions will be performed by the negotiation mechanism. Finally, the function returns four parameters.

- Observation: Offer proposed by opponent and current relative time.
- Reward: Utility value of the current offer and extra reward when an agreement is reached.
- Done: Reach the final state or there is no agreement within the maximum running time.
- Info: State of the negotiation mechanism, extra info used for evaluation.

RESET Resets the environment to an initial state and returns an initial observation, initial observation contains negotiators' initial observation and other information relative to the definition of observation space. Reset the time and current step. Create a new negotiation mechanism session.

RENDER This application is not required because there is no visual output.

CLOSE This application is not needed because there is no need to save the data created by the environment.

SEED Sets the seed for this env's random number generator(s), such as negotiation mechanism.

4.1.4 Game

In addition to implementing the official OpenAI Gym Env interface, class Game is designed to control the entire negotiation mechanism. The purpose of this design is to reduce the modification of negotiation mechanism of NegMAS. In this class, there are some parameters, which are received from the mechanism of NegMAS and passed to the RL algorithms as additional information. The two main methods are defined below.

STEP Checks the state of Game, runs the negotiation mechanism for one step.

STEP_FORWARD Sets the key logic for the running of the negotiation mechanism, because negotiator can learn different strategy in SBE.

4.1.5 Challenges of the environment

OpenAI Gym provides a unified interface for custom environment. But it has some problems, which cannot be directly solved by the interface. These problems occurred during environmental design and will be listed and discussed in the following sections.

Design of Action Space and Observation Space

One relevant consideration is related to RL. In [Bak+19], the author study a modular RL. based on BOA (Bidding strategy, Opponent model and Acceptance condition) framework which is an extension of the work done in [Bak+19]. This framework of RLBOA implements an agent that uses tabular Q-learning to learn the bidding strategy by discretizing the continuous state/action space (not an optimal solution for large state/action spaces as it may lead to curse of dimensionality and cause the loss of relevant information about the state/action domain structure too) [Bag+20]. Compared with tabular Q-learning, deep reinforcement learning algorithms use neural networks to solve this problem.

There are two possible approaches to implementing deep reinforcement learning for this learning case:

The first method: The output size of neural network is directly related to the size of the action space, in other words, it is related to the size of negotiation issues.

The second method: Discrete action space replaced by continuous action space. Before apply the action, filter invalid actions and scale valid actions.

Design of Reward Function

The reward function is the focus of the implementation of the RL. algorithm. It is easy to understand, RL. learns strategies by evaluating the value of actions. Therefore, it is very necessary to design a good reward function. In SBE, the utility function defined by *Negotiator* can be used as a calculation tool to get the current offer reward, which can be intuitively set as part of the reward function.

4.1.6 Analysis of the reinforcement learning algorithms

Policy Optimization vs. Q-Learning

PPO vs. DQN

4.1.7 Conclusion

4.2 SCML with OpenAI Gym

4.2.1 Configuration

Negotiation Issues

Standard SCML Negotiation issues are multi-issues, Quantity, Time and Price.

SCML-OneShot Negotiation issues are multi-issues, Quantity and Price. Time is not important in this simulation world. All contracts will be executed at the same step in which agents reach agreements.

4.2.2 Model

The model consists of six parts, environment MCBE, Scenario, World, Agent, Interface and MADRL algorithms. All six parts are needed to be rewritten according to SCML and OpenAI Gym.

Entire model is shown in 4.2

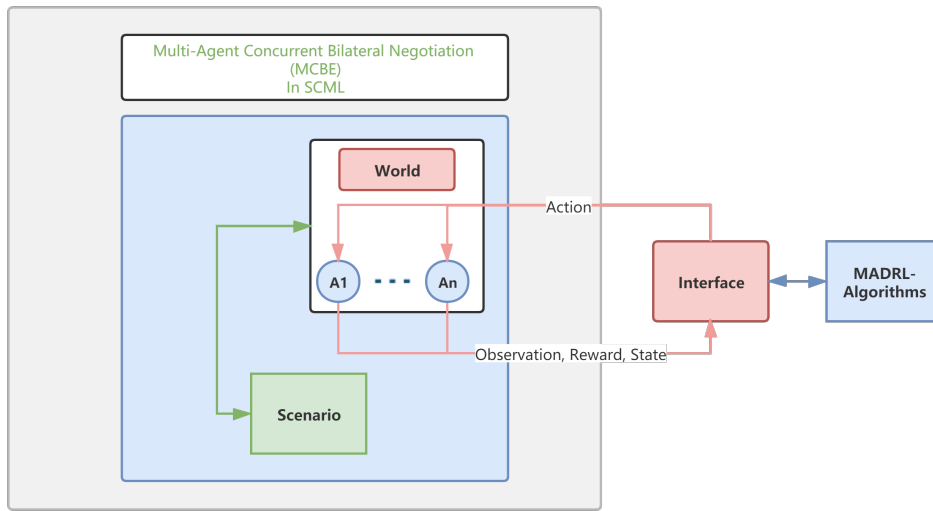


Figure 4.2: Model for Multi-Agent Concurrent Bilateral Negotiation based on SCML

4.2.3 Multi-Agent Environment

In order to be able to realize deep reinforcement learning for multi-agent with an OpenAI Gym Environment, the interface would have to be expanded. In the following, alternative possibilities for using an OpenAI Gym Environment for MARL are discussed. Since MCBE realizes OpenAI Gym env interface methods, a new method named run is added to execute entire episode.

STEP Runs the simulated world for one step. Not important in this case.

RESET Resets the environment(MCBE) and other related parameters to an initial state after every episode and returns an initial observation.

RENDER This application is not required because there is no visual output.

CLOSE This application is not needed because there is no need to save the data created by the environment.

SEED Sets the seed for this env's random number generator(s)

RUN Runs entire episode. After a negotiation step, the rewards, observations, actions, etc. are stored in the memory buffer.

- Observation: Current offer in negotiation mechanism. The observations of all agents are combined in one list. Agent can only access to its local observation during decentralised execution.
- Reward: Sum reward of all learnable agents. Reward of single agent is the sum of utility value of the current offer after one negotiation step and profit of agent after one simulation step.
- Done: Reaches the final state (last step of simulation world) or the maximum running time.
- State: State of environment. It can be replaced by Observation.

4.2.4 Scenario

Scenario describes the structure of simulation world. It is similar as the assisted method Game in SBE and provides logic for generating and resetting the world. With the help of Scenario, many scenarios can be created without changing of MCBE. Figure 4.3 diagrams a simple scenario.

Scenario Interface consists three normal functions and four callback functions passed to MCBE.

MAKE_WORLD Creates instance of game or training world.

RESET_WORLD Sets the world to the initial state.

RESET_AGENT Resets agent, returns initial observation.



Figure 4.3: Example of supply chain scenario, MyOneShotBasedAgent vs. GreedyOneShotAgent

CALLBACK OBSERVATION, REWARD, DONE and INFO

4.2.5 Challenges of the environment

Combination with SCML

Compared with SBE, MCBE can not directly call the function designed in official SCML. Step in SCML-OneShot is one simulation step. In one simulation step, many negotiation steps will be performed. The action of the agent is a proposal, so it needs to be meticulous to control every step of the negotiation mechanism in the simulation world. The class `TrainWorld` inherited from `SCMLOneShotWorld` achieves this goal.

4.2.6 Analysis of the reinforcement learning algorithms

Independent Learning vs. Centralized Learning

Non-stationary environment Traditional reinforcement learning approaches such as Q-Learning or policy gradient are poorly suited to multi-agent environments. One issue is that each agent's policy is changing as training progresses, and the environment becomes non-stationary from the perspective of any individual agent (in a way that is not explainable by changes in the agent's own policy).

MADDPG vs. QMIX

Centralised learning of joint actions can naturally handle coordination problems and avoids non-stationarity, but is hard to scale, as the joint action space grows exponentially in the number of agents. In [Ras+18], author proposed a neural network to transform the centralised state into the weights of another neural network. This second neural network is constrained

to be monotonic with respect to its inputs by keeping its weights positive. This feature makes it possible to learn when there are many agents.

4.2.7 Conclusion

5 Methods and Experiments

5.1 Single-Agent Bilateral Negotiation Environment (SBE)

In this environment, agent represents the negotiator in negotiation mechanism.

5.1.1 Independent Negotiator in NegMAS

In the environment has just single learnable DRL negotiator. All RL. algorithms with discrete action space can be tested in this specific environment. In the experiment of this thesis, DQN and PPO were tested in four learning cases:

- single issue, acceptance strategy
- single issue, offer strategy
- multi-issues, acceptance strategy
- multi-issues, offer strategy

The training step of DQN is shown in Figure 5.1

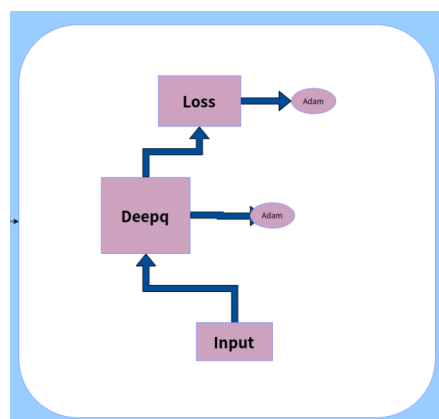


Figure 5.1: Training step of DQN

5.1.2 Experiment

Figure 5.2 diagrams the Game in SBE

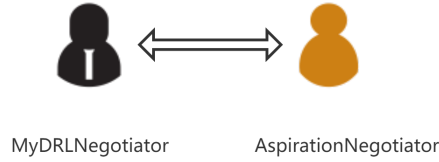


Figure 5.2: Bilateral Negotiation Game in SBE, My Deep Reinforcement Learning Negotiator vs. Aspiration Negotiator

single issue

Negotiation mechanism is SAOM, split the learning strategy as two parts, acceptance strategy and offer strategy,

Acceptance strategy: actions of agent are `Accept offer`, `Wait` and `Reject offer`. Observation of agent are offer of opponent and current time (running time, or current step of negotiation). Algorithms are DQN (blue) and PPO (red). Mean episode reward is shown in 5.3

Offer strategy: Actions of negotiator are all outcomes set in mechanism. The observation is same as defined in the acceptance strategy. Before training the agent, normalize action and observation. Algorithm is DQN and PPO.

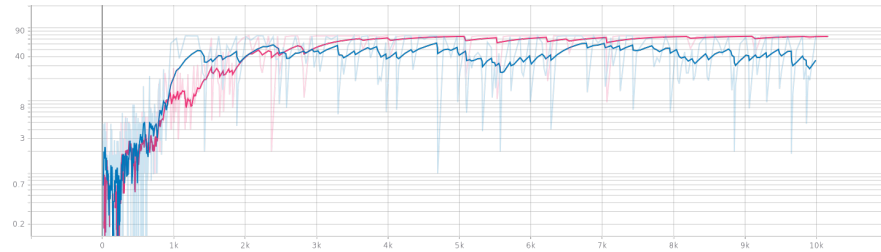


Figure 5.3: Episode mean reward of acceptance strategy under single issue

multi issues

5.1.3 Evaluation

5.2 Multi-Agent Concurrent Bilateral Negotiation Environment (MCBE)

In this environment, agent represents the factory manager and negotiation controller in standard SCML and SCML OneShot, respectively.

The agent interacting with environment may be have many related trainable agents as the part of learner(e.g. one seller, one buyer) in the model. The detail of interactive logic is shown below in 5.4

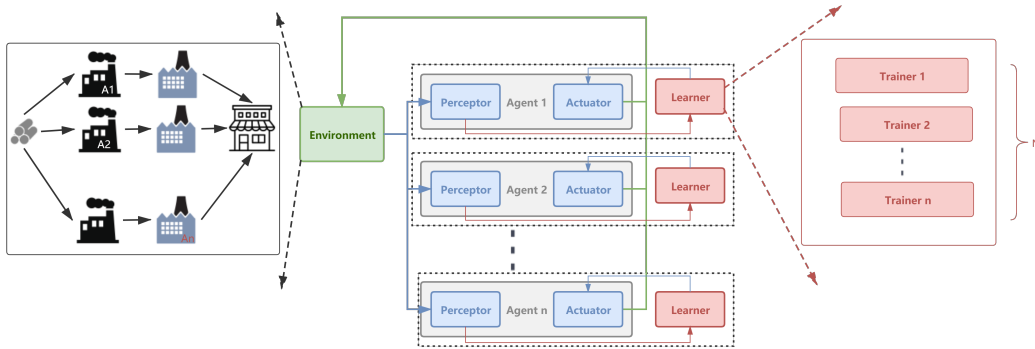


Figure 5.4: Interactive logic based on the perspective of SCML. N: The maximum number of concurrent negotiations for a single agent

5.2.1 MADDPG in SCML

Shown in 5.5

Actors output actions as inputs to related agents interacting with the environment. Agents interacting with environment outputs the observation and reward as the inputs to related agents trained in the model.

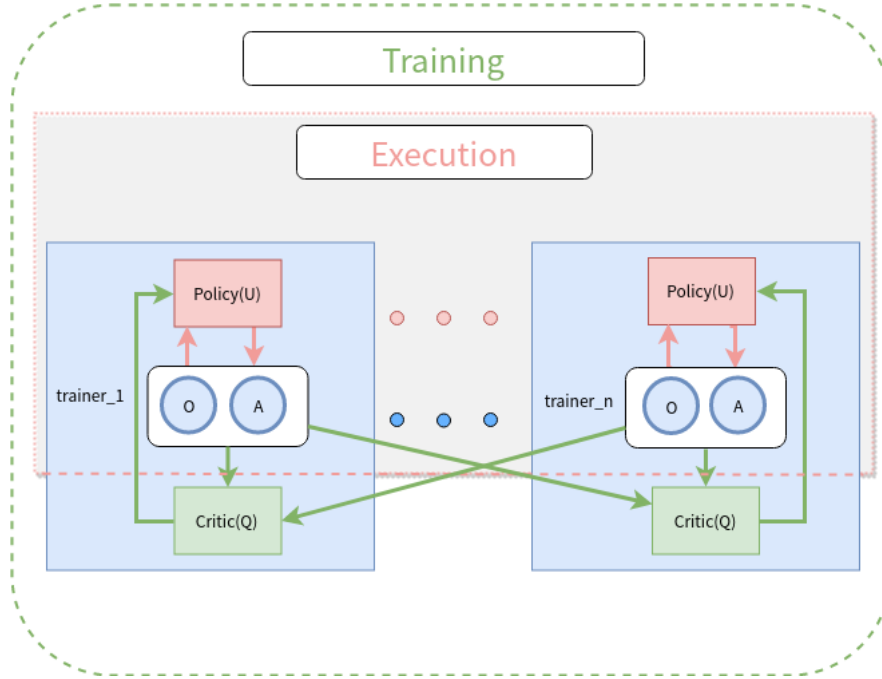


Figure 5.5: MADDPG used in MCBE

Algorithmus 1 : How to write algorithms**Data** : this text**Result** : how to write algorithm with \LaTeX 2e

```

1 initialization;
2 while not at end of this document do
3   read current;
4   if understand then
5     go to next section;
6     current section becomes this one;
7   else
8     go back to the beginning of current section;
9   end
10 end

```

5.2.2 QMIX in SCML

Data flow is shown in 5.6

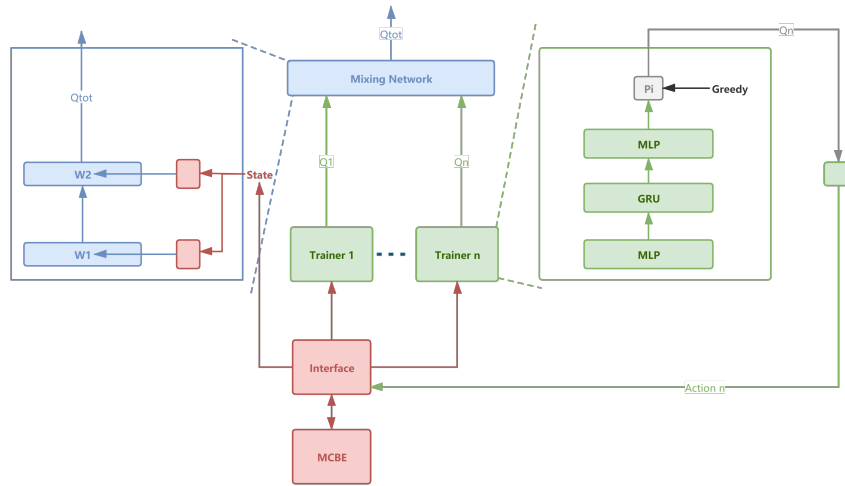


Figure 5.6: QMIX used in MCBE

5.2.3 Experiment

Concurrent Neogtations in standard SCML

Standard SCML is a complex simulation world, which contains various parts with specific functions. The breif description of this simulation is introduced in chapter Background 2.6.3. The experiment of this thesis focus on only the Negotiation Manager of Decision-Maker Agent. The above mentioned method maddpg 5.2.1 is used in this experiment. Scenario is diagramed in Figure 5.7

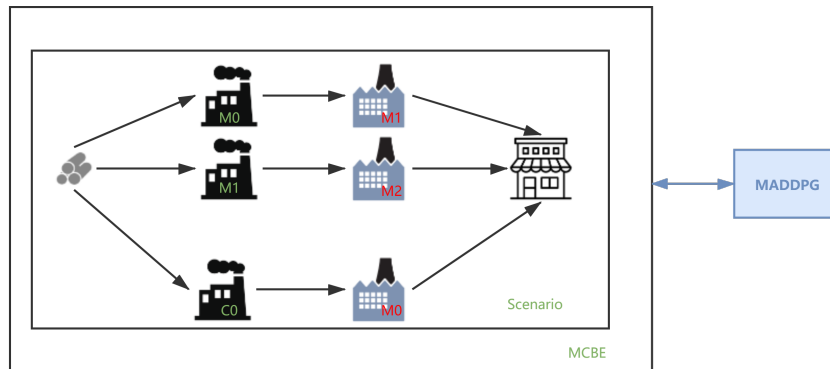


Figure 5.7: M^* represent My Component Based Agent with learner MADDPG, C^* represent Opponent Agents, such as IndDecentralizingAgent

Evaluation Before evaluating the result of concurrent bilateral negotiation with MADDPG in standard SCML. The result of a simple pre-test version which changes the range of negotiation issues with MADDPG is shown in 5.8.

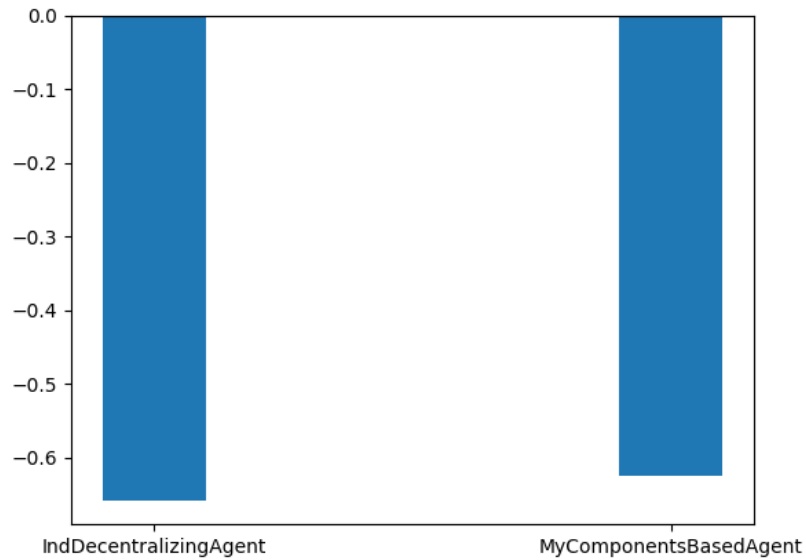


Figure 5.8

Concurrent Negotiations in OneShot SCML

self-play

Episode mean reward curve is shown in 5.9

play with other agent

My agent vs GreedyOneShotAgent

Episode mean reward curve is shown in 5.10

Evaluation

5.3 Conclusion

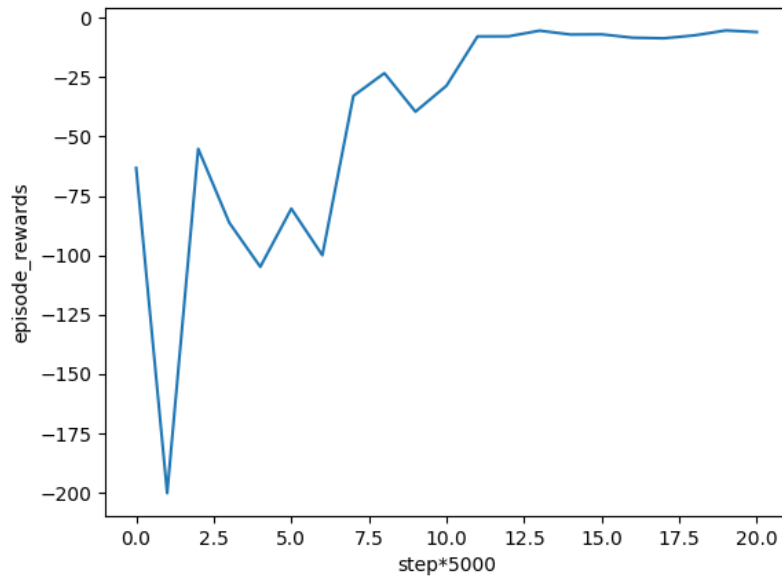


Figure 5.9: Episode mean reward of self paly under SCML OneShot

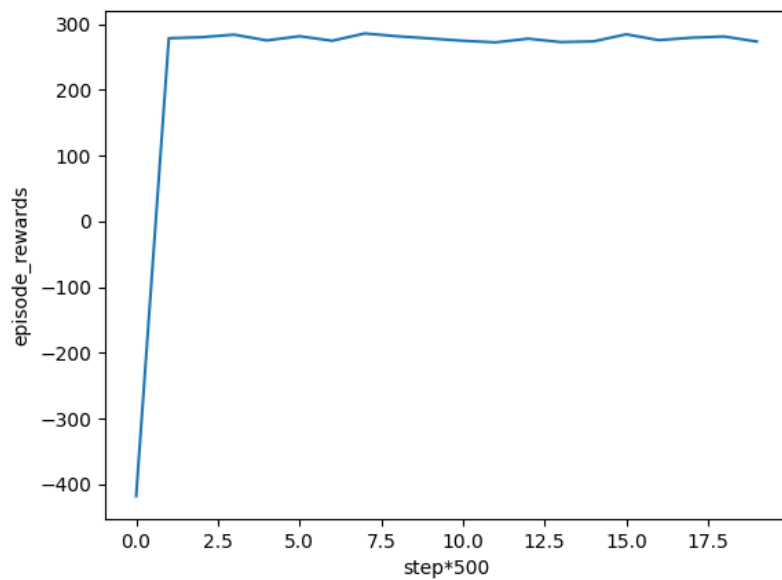


Figure 5.10: Episode mean reward of my agent vs GreedyOneShotAgent under SCML OneShot

6 Conclusions and Future Work

6.1 Others goal

In the SCM league, profit-maximizing is the goal of RL-agent, in game theory we could get many goals, such as welfare-maximization, pareto optimality. How to achieve these goals with RL-methods based on the developed environments in NegMAS and SCML?

6.2 Evaluation

Many metrics in the filed multi agent could be used for evaluating the agents proposed in this paper, such as....

6.3 Design of reward function

Reward function is an important part of realizing of RL-Agent, In the future could develop a more effective reward function, such as method proposed in the paper by [].

6.4 Complex environment

6.5 Huge scale high performance learning

Ray and reverb

Bibliography

- [Bag+20] Pallavi Bagga et al. *A Deep Reinforcement Learning Approach to Concurrent Bilateral Negotiation*. 2020. arXiv: 2001.11785 [cs.MA].
- [Bak+19] Jasper Bakker et al. “RLBOA: A Modular Reinforcement Learning Framework for Autonomous Negotiating Agents.” In: *AAMAS*. 2019.
- [Bro+16] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG].
- [Fan+20] Xiaohan Fang et al. “Multi-Agent Reinforcement Learning Approach for Residential Microgrid Energy Scheduling.” In: *Energies* 13.1 (2020). URL: <https://www.mdpi.com/1996-1073/13/1/123>.
- [Hil+18] Ashley Hill et al. *Stable Baselines*. <https://github.com/hill-a/stable-baselines>. 2018.
- [Lin+14] Raz Lin et al. “Genius: An Integrated Environment for Supporting the Design of Generic Automated Negotiators.” In: *Computational Intelligence* 30 (Feb. 2014), pp. 48–70.
- [Moh+19] Yasser Mohammad et al. “Supply Chain Management World.” In: Oct. 2019, pp. 153–169.
- [Pas+19] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf>.
- [Ras+18] Tabish Rashid et al. “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning.” In: (Mar. 2018).
- [Rub82] Ariel Rubinstein. “Perfect Equilibrium in A Bargaining Model.” In: *Econometrica* 50 (Feb. 1982), pp. 97–109.
- [SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, Second Edition*. MIT Press Cambridge MA, 2018. URL: <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>.

-
- [SUM20] SUMAN. *Is machine learning required for deep learning?* 2020. URL: <https://ai.stackexchange.com/questions/15859/is-machine-learning-required-for-deep-learning>.
- [Wil+12] Colin Williams et al. "Negotiating Concurrently with Unknown Opponents in Complex, Real-Time Domains." In: May 2012.
- [Zho+17] L. Zhou et al. "Multiagent Reinforcement Learning With Sparse Interactions by Negotiation and Knowledge Transfer." In: *IEEE Transactions on Cybernetics* 47.5 (May 2017), pp. 1238–1250.

List of Tables

2.1	stable baselines algorithms	16
-----	---------------------------------------	----

List of Figures

2.1	Sub-areas of artificial intelligence [SUM20]	8
2.2	The agent–environment interaction in reinforcement learning, diagrammed In [SB18].	10
2.3	The multi-agent–environment interaction in reinforcement learning, diagrammed In [Fan+20].	10
3.1	Architecture of the concurrent negotiation agent, best time: t_i and utility value: u_i , probability distributions: P [Wil+12].	18
4.1	Model for single agent bilateral negotiation based on NegMAS	22
4.2	Model for Multi-Agent Concurrent Bilateral Negotiation based on SCML	25
4.3	Example of supply chain scenario, MyOneShotBasedAgent vs. GreedyOneShotAgent	27
5.1	Training step of DQN	29
5.2	Bilateral Negotiation Game in SBE, My Deep Reinforcement Learning Negotiator vs. Aspiration Negotiator	30
5.3	Episode mean reward of acceptance strategy under single issue	30
5.4	Interactive logic based on the perspective of SCML. N: The maximum number of concurrent negotiations for a single agent	31
5.5	MADDPG used in MCBE	32
5.6	QMIX used in MCBE	33
5.7	M* represent My Component Based Agent with learner MADDPG, C* represent Opponent Agents, such as IndDecentralizingAgent	33
5.8	34
5.9	Episode mean reward of self paly under SCML OneShot	35
5.10	Episode mean reward of my agent vs GreedyOneShotAgent under SCML OneShot	35

List of Theorems

2.1	Nash Equilibrium	5
2.2	Pareto Efficient	5

Listings

2.1	Logic of OpenAI Gym Interaction	15
-----	---	----

Glossary

ANAC The International Automated Negotiating Agents Competition (ANAC) is an annual event, held in conjunction with the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), or the International Joint Conference on Artificial Intelligence (IJCAI). The ANAC competition brings together researchers from the negotiation community and provides unique benchmarks for evaluating practical negotiation strategies in multi-issue domains. The competitions have spawned novel research in AI in the field of autonomous agent design which are available to the wider research community. 13

ANEGMA ANEGMA 18

CNS Concurrent negotiation strategy. xii, 17

DQN Deep Q-Value Network 29, 30

DRL Deep reinforcement learning. xi, 12, 21, 29

GreedyOneShotAgent A greedy agent based on OneShotAgent 27, 43

IndDecentralizingAgent Independent Centralizing Agent, implemented in standard scml 33, 43

MADDPG Multi Agent Deep Deterministic Policy Gradient v, xiii, 12, 27, 31–34, 43

MADRL Multi-Agent Deep reinforcement learning. 9, 25

MARL Multi-Agent Reinforcement Learning. 9, 25

MCBE Multi-agent concurrent bilateral negotiation environment. xiii, 21, 25–27, 31–33, 43

MDPs Markov decision process. 5

MyOneShotBasedAgent My Deep Reinforcement Learning Agent in SCM-ONESHOT 27, 43

NegMAS NEGotiation MultiAgent System xi, xii, 13, 14, 21, 23

NegoSI negotiation-based MARL with sparse interactions (NegoSI) 18

OpenAI Gym OpenAI Gym is a toolkit for reinforcement learning research. It includes a growing collection of benchmark problems that expose a common interface, and a website where people can share their results and compare the performance of algorithms. xii, 14, 16, 21, 23–25, 27, 47

PCA Principal Component Analysis.

PDF Portable Document Format.

PG Policy Gradient xi, 12

PPO Proximal Policy Optimization 12, 29, 30

PyTorch Python machine learning framework, developed by... xii, 14

QMIX Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning v, xiii, 13, 27, 32, 33, 43

Ray Ray provides a simple, universal API for building distributed applications. xii, 16

RL. Reinforcement learning. 3, 17, 21–24, 29

RLBOA RLBOA 18, 24

SAOM Stacked Alternating Offers Protocol, namely in SCML also as Stacked Alternating Offers Mechanism. 7, 13, 21, 22, 30

SBE Single-agent bilateral negotiation environment. xii, 21–24, 26, 27, 29, 30, 43

SCM Supply Chain Management 1, 13, 14

SCML Supply Chain Management League one of ANAC 2020 and 2021 leagues @ IJCAI 2020 and 2021. xi–xiii, 1, 6, 13, 14, 21, 24, 25, 27, 31–34, 43

SCML-OneShot OneShot World in SCML. 25, 27