Multiagent Reinforcement Learning and Heterogeneity

Bachelor's Thesis of

Christian Burmeister

at the Department of Informatics
Institute for Anthropomatics and Robotics (IAR)
Autonomous Learning Robots (ALR)

Reviewer:	Prof. A
Second reviewer:	Prof. B
Advisor:	M.Sc. C
Second advisor:	M.Sc. D

xx. Month 2021 – xx. Month 2021

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . .

(Christian Burmeister)

Abstract

asdf

Zusammenfassung

Contents

List of Figures

List of Tables

1. Motivation

1.1. Multiagent Systems

Use Cases:
- Multiagent systems can be used in game theory and financing
- Reconnaissance robots covering a wide area. Communication not always possible.
- Smart Grid for Electricity, Power allocation, energy management.
- Flow Line Systems
- Stock markets
- Competitive pricing strategies
- Load Balancing.
- Network Systems (IoT).
- Traffic Light Control
- Autonomous Driving, Vehicular networks
- Automating turbulence modelling (aircraft design, weather forecasting, climate prediction).
- Control Systems for industrial processes.
- Intrusion Detection
- resource allocation for UAV Networks
- Large Scale City Traffic (Cityflow).
- Spectrum Management of cognitive radio using MARL.

Aspects:
- Ant-Colony-Optimization, which can be used for learning.
- Emergent Behavior.
- Swarm Intelligence.
- multi-agent reinforcement learning.
- multi-agent learning.
- game theory

This is the SDQ thesis template. For more information on the formatting of theses at SDQ, please refer to `https://sdqweb.ipd.kit.edu/wiki/Ausarbeitungshinweise` or to your advisor.

1.2. Spacing and indentation

To separate parts of text in LaTeX, please use two line breaks. They will then be set with correct indentation. Do *not* use:

- `\\`
- `\parskip`

Figure 1.1.: SDQ logo

abc	def
ghi	jkl
123	456
789	0AB

Table 1.1.: A table

- `\vskip`

or other commands to manually insert spaces, since they break the layout of this template.

1.3. Example: Citation

A citation: [1]

1.4. Example: Figures

A reference: The SDQ logo is displayed in Figure 1.1. (Use `\autoref{}` for easy referencing.)

1.5. Example: Tables

The `booktabs` package offers nicely typeset tables, as in Table 1.1.

1.6. Example: Formula

One of the nice things about the Linux Libertine font is that it comes with a math mode package.

$$f(x) = \Omega(g(x)) \ (x \to \infty) \ \Leftrightarrow \ \limsup_{x \to \infty} \left| \frac{f(x)}{g(x)} \right| > 0$$

2. Information to sort

2.1. MARL - A Comprehensive Survey of Multiagent Reinforcement Learning - 2008

MARL - A Comprehensive Survey of Multiagent Reinforcement Learning - 2008
Benefits

- can be parallelized.
- can use experience sharing via communication, or with a teacher-learner relationship.
- Failure of one agent can be covered by other agents.
- insertion of new agents => scaleable.
- MARL Complexity: Exponential in number of agents.
- exploration (new knowledge) - exploitation (current knowledge) - Tradeoff.
- They explore about the environment and other agents.
- need for coordination.

Application Domains

- simulation better than real-life (better scalability and robustness).
- Distributed Control: for controlling processes (for larger industry plants).
  - avenue for future work.
  - used for traffic, power or sensory networks.
  - could also be used for pendulum systems.
- Robotic Teams (Multirobot):
  - simulated 2D space.
  - navigation: Reach a goal with obstacles. Area sweeping (retrival of objects (also cooperative)).
  - pursuit: Capture a prey robot.
- Automated Trading: Exchange goods on electronic markets with negotiation and auctions.
- Resource Management: Cooperatie team to manage resources or as clients. (routing, load balancing).

Practicallity and Future works

- Scalability Problem: Q-functions do not scale well with the size of the state-action space.
  - Approximation needed: for discrete large state-action spaces, for continous states and discrete actions or continious state and action.
  - Heuristic in nature and only work in a narrow set of problems.
  - Could use theoretical results on single-agent approximate RL.
  - also could use discovery and exploitation of the decentralized, modular structure of the multiagent task.
- MARL without prior knowledge is very slow.
  - Need humans to teach the agent.

  - shaping: first simple task then scale them.
  - could use reflex behavior.
  - Knowledge about the task structure.
- Incomplete, uncertain state measurements could be handled with partiall observability techniques (Markov decision process).
- Multiagent Goals needs a stable learning process for the environment and an adaption for the dynamics of other agents.
- using game-theory-based analysis to apply to the dynamics of the environment.

2.2. !MAS - An Introduction to Multi-Agent Systems - 2010

MAS - An Introduction to Multi-Agent Systems - 2010

Benefits of using MAS in large systems

- Increase in the speed and efficiency of the operation due to parallel computation and asynchronous operation.
- Graceful degradation whone one or more of the agent fail, thus increasing realibility and robustness of the system.
- Scalability and flexibility - Agents can be added as and when necessary.
- Cost Reduction: Individual agents cost much less than a centralized architecture
- Reusability: Agents with a modular structure can be easily replaced in other systems or be upgraded more easily than a monolithic sysetm.

Challenges of using MAS in large systems

- environment: An agents action modify its own environment but also that of its neighbours. therefore they need to predict the action of the other agents so that they can reach a goal. This can be an unstable system. Environment dynamic: Is the effect caused by other agents or by the variation in the environment?
- perception: limited sensing range => each agent only has partial observability for the environment. Therefore the decisions reached might be sub-optimal.
- Abstraction: ???
- conflict resolution: lack of global view => conflict. therefore information on constraints, action preferences and goal prioritoes must be shared between agents. When to communicate what to which agent?
- Inference: Single-Agent: State-Action-Space can be mapped with trial and error. Multiagent: each agent may or may not interact with each other. If they are heterogenous, they might even compete and have different goals. You need a fitting inference machanism

2.2.1. Classification of MAS

Internal Architecture

- homogeneous: all agents have the same internal architecture (Local Goals, Sensor Capabilities, Internal states, Inference Mechanism and Possible Actions). In a typical distributed environment, overlap of sensory inputs is rarely present

- Heterogeneous: agents may differ in ability, structure and functionality. Because of the dynamics and location the actions chosen might differ between agents. their local goals may contradict the objective of other agents.

Agent Organization

- hierarchical: typical: tree-structure. At different heights, different levels of autonomy. data from lower levels flow upwards. Control signal flows from high to low in the hierarchy.
  - simple: the decision making authority is a single agent of highest level. BUT: single point of Failure
  - uniform: authority is distributed among the various agents, for better efficiency, fault tolerance, graceful degradation. Decisions made by agent with appropriate information. (MAS - TrafficControl - Neural Networks for Continuous Online Learning and Control - 2006)
- holonic: fractal structure of several holons. Self-repeating. Used for large organizational behaviours in manufacturing and business.
  - An agent that appears as a single entity might be composed of many sub-agents. They are not predetermined, but form through commitments.
  - Each holon has a head agent that communicates with the environment or with other agents in the environment. It is selected either randomly, through a rotation policy, or selected by resource availability, communicaton capability.
  - Holons can be nested to form Superholons.
  - compare to tree: in Holons cross tree interactions and overlapping of holons is allowed.
  - pro: abstraction good degree of freedom, good agent autonomy.
  - contra: abstraction makes it difficult for other agents to predict the resulting actions of the holon.
- coalitions: group of agents come together for a short time to increase utility or performance of the individual agents in a group. they cease to exist when the performance goal is achieved.
  - coalition may have either a flat or a hierarchical architecture.
  - It may have an leading agent to act as a representative.
  - overlap is allowed. this increased complexity of computation of the negotiation strategy.
  - You can have one coaltion with all agents => maximum performance of system. Impractical due to restraints on communication and resources.
  - minimize amount of colations: because of the cost of creating and dissolving a colation group.
- teams: agents work together to increase the overall performance of the group, rather than working as individual agents.
  - their interactions can be arbitrary and the goals and roals can vary with the performance of the group.
  - large team size is not beneficial under all conditions. some compromises must be made.
  - large teams offer a better visibility of the environment. but is slower computation wise. Learning-Performance Tradeoff.

  - computation cost usually much greater than coalitions.

Communication

- local communication: agents directly communicate similar to message passing. there is no place to store information. creates distributed architecture. used in: (25),(37),(38).
- blackboards: a group of agents share a data repository which is provided for efficient storage.
  - can hold design data and control knowledge, accessable by the agents.
  - control shell: notfies the agent when relevant data is available.
  - single point of failure.
- agent communication language (ACL): common framework for interaction and information sharing. (40).
  - procedural approach: modelled as a sharing of the precedural directives. Shared how an agent does a specific task or the entire working of the agent itself. Script Languages often used. Disadvantage: necessitiy of providing information on the recipient agent, which is in most cases partially known. Also how to merge the scripts into one executable. Not preferred method.
  - declarative approach: sharing of statements for definitions, assumptions assertions, axioms etc. Short declarative statements as length increases probability of information corruption. Example: ARPA knowledge sharing effort.
  - Best known inner languages: Knowledge Interchange Format. Information exchange is implicitly embedded in KIF. But the package size grows with the increase in embedded information. Solution: High-level Languages like KQML (Knowledge QUery and Manipulation Language)

Decision making in Multi-Agent Systems

- undercainty: effects of a specific actions on the environment and dynamics because of the other agents.
- Methodology to try and find a joint action or equilibrium point which maximizes the reward of every agent.
- Typically modelled with game theory method. Strategic games:
  - a set of players (agents)
  - Foreach player, there is a set of actions
  - Foreach player, the prefeernces over a set of actions profiles
  - payoff with the combination of action, a joint-action, that is assumed to be predefined.
  - all actions are observable forall agents.
  - make the assumption that all participating agents are rational.
- Nash equilibrium: for a payoff matrix: An action profile (joint-action), where no player can do better by choosing one of the actions differently, given that the other player chose a specific action.
- there might be multiple nash equilibrium, so that there is no dominant solution. Here the coordination of MAS is needed to find a solution.
- Iterated Elimination Method: Strongly dominated actions are iteratively eliminated. This fails if there are no strictly dominated actions available.

Coordination

- agents work in parallel, therefore they need to be coordinated or synchronize the actions to ensure stability of the system.

- other reasons: prevent chaos, meet global constraints, utilize distributed resources, prevent conflicts, improve efficiency.
- achievable with constraints on the joint actions or by using informatil collated from neighbouring agents. Used to find the equilibrium action.
- payoff matrix necessary might be difficult to determine. It increases expenentially in the number of agents and action choices.
- dividing the game into subgames: roles (permitted actions is reduced, good for distributed coordination or centralized coordination)
- Coordination via Protocol.
  - negotioation to arrive an approdiate solutions.
  - Agents assume the role of manager (divide the problem) and contractor (who deals with the subproblems).
  - The manager and contractor are working in a bidding system.
  - Example: FIPA model
  - disadvantage: assumption of the existence of an cooperative agent. It is very communication intensive
- Coordination via Graphs: Problem is subdivided into easer problems. Assume the payoffs can be linear combinated from the local payoffs of the sub-games. THen just eliminate agents to find the optimal joint.
- Can also use belif models. Internal models of an agent on how he believes the environment works (needs to differentiate between environment and effects of other agents).

Learning

- active learning: analysing the observations to creat a belief or internal model of the corresponding situated agent's environment.
  - can be performed by using a deductive, inductive or probabilistic reasoning approach.
  - deductive: inference to explain an instance or state-action sequence using his knowledge. It is deduced or inferred from the original knowledge it is nothing new. It could form new parts of the knowledge base. uncertainty is usually disregarded (not good for real-time)
  - inductive: learning from observations of state-action pair. Good when environment can be presented in terms of some generalized statements. they use the correlation between observations and the action space.
  - probabilistic: assumption: knowledge base or belief model can be represented as probabilities of occurrence of events. observations of the environment is used to predict the internal state of the agent. Good example: Bayesian learning. Difficult for MAS, as the joint probability scales poorly in the number of agents.
- reactive learning: updating belief without having the actual knowledge of what needs to be learnt.
  - useful when the underlying model of the agent or the environment is not known clearly and are black boxes.
  - can be ssen in agents which utilize connections systems such as NN.
  - can use reactive multi-agent feed forward neural networks.
  - they depend on the application domain and are therefore rarely employed in real world scenarios.
- learning based on consequences:

  - learning methods based on evaluation of the goodness of selected action. like in reinforcement learning.
  - programming the agents using reward and punishment scalar signals without specifying how the task is to be achieved.
  - learnt through trial and error and interaction with the environment.
  - usually used when action space is small and descret. Recent developments allow them to work in continious and large state-action space scenarios.
  - An agent is usally represented as a Markov Decision Process.
  - Expectaation operator optmal policy is the argmax of the Q-value, which uses the bellman equation. Bellman equation is solved iteratively.
  - The solution is referred to as q-learning method.
  - For MAS the reinforcement learning method has the problem of combinatorial explosion in the state-action pairs.
  - The information must be passed between the agents for effective learning.

2.3. Artificial Intelligence - A modern Approach

2.3.1. Agents and Environments

p.34

- **agent**: anything that perceives its **environment** through **sensors** and acting upon that environment using **actuators**.
- **percept**: agent's perceptual inputs at any given instance. Percept sequence is a complete history of perception.
- agents choice of action decided upon the history of perception, but not anything it has not perceived.
- its behavior is described by the **agent function**, which is internally implemented by the **agent program**.

2.3.2. Rational Agent

p.36

- **rational agent**: it does the correct thing. Correctness is determined by a performance measure, which is determined by the changed environment states.
- design **performance measures** according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.
- rational depends on:
  - the performance measure that defines the criterion of success
  - the agent's prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent's percept sequence of data.
- depending on the measures the agent might be rational or not.

- an **omniscient agent** knows the actual outcome of its actions and can act accordingly, but this is impossible in reality.
- rationality maximizes expected performance, while perfection (omniscient) maximizes actual performance.
- agents can do actions in order to modify future percepts, called **information gathering, or exploration**.
- rational agents learn as much as possible from what it perceives.
- his knowledge can be augmented and modified as it gains experience.
- if the agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks **autonomy**.
- it should learn what it can to compensate for partial or incorrect prior knowledge.
- give it some initial knowledge and the ability to learn, so it will become independent of its prior knowledge.

2.3.3. Nature of Environments

p.40

- **task environments**: the "problems" to which rational agents are the "solutions".
- Describe the task environment in the following aspects P(Performance measure), E(Environment), A(Actuators), S(Sensors).
- **fully observable**: the agent's sensors give it access to the complete state of the environment. All aspects that are relevant to the choice of actions
- **partially observable**: otherwise. Because of missing sensors or noise.
- no sensors: unobservable
- single-agent environments and multi-agent environments.
- multi-agent can be either competitive (chess) or cooperative (avoiding collisions maximizes performance).
- **communication** emerges as a rational behavior in multiagent environments.
- randomized behavior is rational because it avoids the pitfalls of predictability.
- **Deterministic**: next state of environment is completely determined by the current state and the action executed by the agent, otherwise it is **stochastic**.
- you can ignore uncertainty that arises purely from the actions of other agents in a multiagent environment.
- If the environment is partial observable, it could appear to be stochastic, which implies quantifiable outcomes in terms of probabilities.
- an environment is **uncertain** if it is not fully observable or not deterministic.
- **episodic**: the agent's experience is divided into atomic episodes. In each the agent receives a percept and performs a single episode. The next episode does not depend on the actions taken in previous episodes, otherwise it is **sequential**.
- When the environment can change while the agent is deliberating, then the environment is **dynamic** for that agent otherwise it is **static**.
- if the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **semi dynamic**.
- **discrete/continuous** applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agents.

- **known vs. unknown**: refers to the agent's state of knowledge about the "laws of physics" of the environment. Known environment, the outcomes for all actions are given, otherwise the agent needs to learn how it works. An environment can be known, but partially observable (solitaire: I know the rules but still unable to see the cards that have not yet been turned over)
- hardest case: partially observable, multiagent, stochastic, sequential, dynamic, continuous, and unknown
- **environment class**: multiple environment scenarios to train it for multiple situations.
- you can create an **environment generator**, that selects environments in which to run the agent.

2.3.4. Structure of Agents

p.46

- agent = architecture (computing device) + program (agent program).
- agent programs take the current percept as input and return an action to the actuators.
- agent program takes the current percept, agent function which takes the entire percept history.
- **table driven agent**: Uses a table of actions indexed by percept sequences. This table grows way to fast and is therefore not practical.

Simple reflex agents:

- **simple reflex agents**: Select the actions on the basis of the current percept, ignoring the rest of the history.
- **condition-action-rule**: these agents create actions in a specific condition (if-then). These connections can be seen as reflexes.
- uses an **interpret-input** function as well as a **rule-match** function.
- they need the environment to be fully observable. They could run into infinite loops.
- you can mitigate this by using randomization for the actions. Which is non-rational for single agent environments.

Model-based reflex agents:

- keep track of the part of the world an agent cannot see now. It maintains some sort of **internal state** that depends on the percept history.
- agents needs to know how the world evolves independently of the agent and how the agent's own actions affect the world.
- with this it creates a **model** of the world hence it is called model-based agent.
- it needs to update this state given sensor data.
- this model is a **best guess** and does not determine the entire current state of the environment exactly.

Goal-based agents:

- an agent needs some sort of **goal information** that describes situations that are desirable. This can also be combined with the model.
- Usually agents need to do multiple actions to fulfill a goal which requires **search** and **planning**.
- this also involves consideration of the future.

- the goal-based agent's behavior can be easily changed to go to a different destination by using a goal where a reflex agent needs completely now rules.

Utility-based agents:
- goals provide a crude binary distinction between good and bad states.
- use an internal **utility function** to create a performance measure.
- if the external performance measure and the internal utility function agree, the agent will act rationally.
- if you have conflicting goals the utility function can specify the appropriate **tradeoff**.
- if multiple goals cannot be achieved with certainty, utility provides a way to determine the **likelihood** of success.
- a rational utility-based agent chooses the action that **maximizes the expected utility**.
- any rational agent must behave as if it possesses a utility function whose expected value it tries to maximize.
- a utility-based agent must model and keep track of its environment.

Learning Agents:
- it allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.
- 4 conceptual components: **learning element** (responsible for improvements), **performance element** (select external action), **critic** (gives feedback to change the learning element), **problem generator** (suggesting actions that lead to new and informative experiences).
- critic tells the learning element how well the agent is doing given a performance standard. It tells the agent which percepts are good and which are bad.
- problem generator allows for exploration and suboptimal actions to discover better actions in the long run.
- learning element: simplest case: learning directly from the percept sequence.
- the **performance standard** distinguishes part of the incoming percept as a reward or penalty that provides direct feedback on the quality of the agent's behavior.

How the components of agent programs work:
- **atomic representation**: Each state of the world is indivisible. Algorithms like search and game-playing, Hidden Markov models and Markov decision models work like this.
- **factored representation**: splits up each state of a fixed set of variables or attributes which each can have a value. Used in constraint satisfaction algorithms, propositional logic, planning, Bayesian networks.
- **structured representation**: here the different states have connections to each other. Used in relational databases, first-order logic, first-order probability models, knowledge-based learning and natural language understanding.
- more complex representations are more **expressive** and can capture everything more concise.

2.3.5. Multiagent Planning

p.425
- each agent tries to achieve is own goals with the help or hindrance of others

- wide degree of problems with various degrees of **decomposition of the monolithic agent**.
- multiple concurrent effectors => **multieffector planning** (like type and speaking at the same time).
- effectors are physically decoupled => **multibody planning**.
- if relevant sensor information foreach body can be pooled centrally or in each body like single-agent problem.
- When communication constraint does not allow that: **decentralized planning problem**. planning phase is centralized, but execution phase is at least partially decoupled.
- single entity is doing the planning: one goal, that every body shares.
- When bodies do their own planning, they may share identical goals.
- **multibody**: centralized planning and execution send to each.
- **multiagent**: decentralized local planning, with coordination needed so they do not do the same thing.
- Usage of **incentives** (like salaries) so that goals of the central-planner and the individual align.

Multiple simultaneous actions:

- **correct plan**: if executed by the actors, achieves the goal. Though multiagent might not agree to execute any particular plan.
- **joint action**: An Action for each actor defined => joint planning problem with branching factor bn̂ (b = number of choices).
- if the actors are **loosely coupled** you can describe the system so that the problem complexity only scales linearly.
- standard approach: pretend the problems are completely decoupled and then fix up the interactions.
- **concurrent action list**: which actions must or most not be executed concurrently. (only one at a time)

Multiple agents: cooperation and coordination

- each agent makes its own plan. Assume goals and knowledge base are shared.
- They **might choose different plans** and therefore collectively not achieve the common goal.
- **convention**: A constraint on the selection of joint plans. (cars: do not collide is achieved by "stay on the right side of the road").
- widespread conventions: social laws.
- absence of convention: use communication to achieve common knowledge of a feasible joint plan.
- The agents can try to **recognize the plan other agents want to execute** and therefore use plan recognition to find the correct plan. This only works if it is unambiguously.
- an **ant** chooses its role according to the local conditions it observes.
- ants have a convention on the importance of roles.
- ants have some learning mechanism: a colony learns to make more successful and prudent actions over the course of its decades-long life, even though individual ants live only about a year.
- Another Example: **Boid**

- If all the boids execute their policies, the flock inhibits the emergent behavior of flying as a pseudorigid body with roughly constant density that does not disperse over time.
- **most difficult multiagent** problems involve both cooperation with members of one's own team and competition against members of opposing teams, all without centralized control.

2.3.6. Game Theory

p.666

2.3.7. Mechanism Design for Multiple Agents

p.679

2.3.8. Adversarial Search

p.182

2.3.9. Probabilistic Reasoning over Time

p.587

2.3.10. Reinforcement Learning

p.830

2.3.11. Planning Uncertain Movements (Potential Fields)

p.993

2.4. Ant Colony Optimization

2.4.1. Wikipedia Article

Ant Colony Optimization Algorithm, Wikipedia

- is used for solving computational problems which can be reduced to finding good paths through graphs.
- artificial ants locate optimal soluions by moving through a parameter space represneting all possible solutions.
- they record their positions and the quality of their solutions for later iterations to find better solutions (pheromones).

2.5. UNSORTED

Gordon 2000: Ants at Work.
Gordon 2007: Control without hierarchy. Nature.
Links:

Ant Simulation Video 1
Ant Simulation Video 2
Boids Video
Distributed Artificial Intelligence, Wikipedia
Multi-agent learning, Wikipedia
Bees algorithm, Wikipedia
Swarm Intelligence, Wikipedia

2.6. References & Papers

2.6.1. Ant Colony Optimization (ACO)

ACO - Ant Colony Optimization for learning Bayesian network - 2002

2.6.2. Multi Agent Reinforcement Learning (MARL)

MARL - Multiagent Reinforcement Learning - Theoretical Framework and an Algorithm - 1998
MARL - Deep Reinforcement Learning for Robot Swarms - 2019 - KIT
MARL - Hierarchical multi-agent reinforcement learning - 2006
MARL - Learning to Communicate with Deep Multi-Agent Reinforcement Learning - 2016
MARL - Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning - 2017
MARL - GAMA - Graph Attention Multi-agent reinforcement learning algorithm for cooperation - 2020
MARL - Plan-based reward shaping for multi-agent reinforcement learning - 2016
MARL - Multi-Agent Reinforcement Learning Using Linear Fuzzy Model Applied to Cooperative Mobile Robots - 2018
MARL - Multi-Agent Reinforcement Learning - a critical survey - 2003
MARL - Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning - 2017
MARL - Mean Field Multi-Agent Reinforcement Learning - 2018
MARL - Multi-Agent Reinforcement Learning A Selective Overview of Theories and Algorithms - 2021
MARL - Coordinating multi-agent reinforcement learning with limited communication - 2013
MARL - Multi-Agent Reinforcement Learning A Report on Challenges and Approaches - 2018
MARL - LIIR - Learning Individual Intrinsic Reward inMulti-Agent Reinforcement Learning - 2019
MARL - PettingZoo - Gym for Multi-Agent Reinforcement Learning - 2020
MARL - Networked Multi-Agent Reinforcement Learning in Continuous Spaces - 2018

MARL - A Review of Cooperative Multi-Agent Deep Reinforcement Learning - 2019
MARL - Transfer Learning in Multi-agent ReinforcementLearning Domains - 2011
MARL - Parallel Transfer Learning in Multi-Agent Systems What, when and how to transfer - 2019
MARL - ROMA Multi-Agent Reinforcement Learning with Emergent Roles - 2020
MARL - A modular approach to multi-agent reinforcement learning - 2005
MARL - Multi-agent reinforcement learning weighting and partitioning - 1999
MAS - Transfer Learning for Multi-agent Coordination - 2011
MARL - Transfer among Agents An Efficient Multiagent Transfer Learning Framework - 2020
MARL - Agents teaching agents a survey on inter-agent transfer learning - 2019
MARL - A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems - 2019

2.6.3. Multiagent Systems (MAS)

MAS - Multi-Agent Systems - A Survey - 2018
MAS - Distributed Cooperative Control and Communication for Multi-agent Systems - 2021
MAS - PRIMA 2020 Principles and Practice of Multi-Agent Systems - 2021
MAS - The Multiagent Planning Problem - 2016
MAS - Swarm Intelligence - 2010
MAS - Swarm Intelligence - 2012
MAS - Swarm Intelligence - 2014
MAS - Swarm Intelligence - 2016
MAS - Swarm Intelligence - 2018
MAS - Swarm Intelligence - 2020
MAS - Transfer learning in multi-agent systems through paralllel transfer - 2013
MAS - Co-evolutionary Multi-agent System with Predator-Prey Mechanism for Multi-objective Optimization - 2007
MAS - Hierarchical Control in a Multiagent System - 2007
MAS - Holonic - A Taxonomy of Autonomy in Multiagent Organisation - 2003
MAS - A survey of multi-agent organizational paradigms - 2004
MAS - HomoHetero - Multiagent Systems A Survey from a Machine Learning Perspective - 2000

2.6.4. Applications

MAS - TrafficControl - Neural Networks for Continuous Online Learning and Control - 2006

3. Fundamentals

Topics:

- multiagent/multibody Systems (MAS).
  - MAS Reinforcement Learning
    - They use stochastic games (Markov Games) as generalization of Markov Decision Processes.
  - Hierarchical MAS, Hierarchical Reinforcement Learning for MAS.
  - MAS with Cooperation and Competition.
  - Particle Swarms (nicht so meins).
  - Problems:
    - MAS Movement Problems (Potential Fields). Mean fields?
    - ? using MAS for Moving a Multi-Legged Robot (Spider-like) with a navigation problem design as a hierarchical MAS?
    - Path Planning Navigation with Heterogeneous Agents?
    - MAS Task Problems: Rendezvous, Pursuit Evasion (Single and one Evader) (Boid?), (MAS - Deep Reinforcement Learning for Robot Swarms - 2019 - KIT)
    - Multi-Agent Path Finding (MAPF). Scalability for this: For fixed space they get into each others way.
    - Collective Foraging: (Ants-kinda). Problem when communication only happens in an area, use local information exchange groups. Information Transfer. (MAS - Swarm Intelligence - 2020), Preferential Foraging (MAS - Swarm Intelligence - 2018 - p.289)
    - Coverage: Multi-robot Information Gathering / Scouting. (MAS - Swarm Intelligence - 2020), Pattern Formation (MAS - Swarm Intelligence - 2016 - p.14)
    - Coalition: Heterogeneous Group of Agents. They have different skills / attrributes that affect the environment. Like an Ant Caste System? Some Agents have better sensors? Only some agents have some sensors? What if the specialization is taken to the extreme? (MAS - Swarm Intelligence - 2020). Limited Visibility Sensors (MAS - Swarm Intelligence - 2018 - p.56). Going from a homogeneous group of agents, to randomized specialities, to extrem specializations. How does it change? Mixed with an Hierarchical Approach? They need to find groups to work together? Some are fast (but cannot see much, there is an insect that cannot see while running), Some have good sensors. Communication range? Genetic Diversity, Task-Allocation and Task-Switching (MAS - Swarm Intelligence - 2016 - p.109)
    - Collective Gradient Perception: Using Abilities of other Agents to take advantage of the whole group. (Flocking) (MAS - Swarm Intelligence - 2020)
    - Indirect Communication through changing states in the environment (birds transport something via cable). Also like using Pheromone Trails (Quality-

    Sensitive Foraging through virtual pheromone trails). (MAS - Swarm Intelligence - 2018 - p.15 - p.147)
    * Control Architecture: Behavior Trees, FSM. (MAS - Swarm Intelligence - 2018 - p.42)
    * Maze-Like Environment with Ant Algorithms (MAS - Swarm Intelligence - 2018 - p.162)
    * Search and Rescue? (Kinda like Foraging?)
    * Disruption: Disrupting Aspects of the Swarm and how they react to it, Swarm Attack: (MAS - Swarm Intelligence - 2018 - p.225), Coherence of Collective Decision Making (MAS - Swarm Intelligence - 2018 - p.264)
    * Evolutionary Systems: NEAT (MAS - Swarm Intelligence - 2012 - p.98)
- Graph-Based Visualisation for MAS. (MAS - Swarm Intelligence - 2010). How do you visualize them?
- Transfer Learning for MARL/MAS
    * Some approaches for parallel transfor of different problems even for MARL Problems.
    * So you can transfer even in parallel.
    * But they only transfer between similar problems. Which would held if you can create a simpler version of your problem and make it more and more complex.
    * Are there transfer learning approaches for MAS/MARL, so that Learning can be transfered between agents? So that if you add agents the complexity isn't as steep?
- Adaptive Learning for MAS?
- Control System: Either fully self-organizing or completely centralized. Hybrid Control of Swarms (MAS - Swarm Intelligence - 2018 - p.69)
- Simulation of MAS: ARGoS
- Best-of-n Problem: Swarm selects best option out of n alternatives. (MAS - Swarm Intelligence - 2018 - p.251)
- Sensory Errors for Foraging, Dynamic Task Partiotioning (MAS - Swarm Intelligence - 2016 - p.124), Task Partitioning Problem (MAS - Swarm Intelligence - 2012 - p.122)
- Task Hierarchy, Multi-Objective
- Random Walks as a search strategy (MAS - Swarm Intelligence - 2016 - p.196)
- Critic: Centralized Critic or Learning individual intrinsic reward (LIIR)
- Standardizing Testing Scenarios (PettingZoo).
- Role concept to for MAS. Agents with similar role share similar behavior.
- Modular Approach to MARL to remedy the poor scalability in the state-space in the number of partner agents.
- Weighting and Partitioning to decrease complexity.
- Hierarchical Groups of MAS where each epoch each Group (5 Agents) exchange Data for learning. And every 10 Epoch the groups exchange data for learning? Randomize these groups? (every few epoch?). Groups of 5 that get reshuffled every 10 Epoch or so.
- holonic agent structure: fractals structure of MAS. holonic and heterogenous? Does this form naturally for extreme heterogenous?
- holonic coalitions?

4. Problem and Approaches

4.1. Definition of the Problemdomain

5. Project

6. Related Works

Is this part of the motivation???

7. Conclusion

Bibliography

[1] Steffen Becker, Heiko Koziolek, and Ralf Reussner. “The Palladio Component Model for Model-driven Performance Prediction”. In: *Journal of Systems and Software* 82 (2009), pp. 3–22. DOI: 10.1016/j.jss.2008.03.066. URL: http://dx.doi.org/10.1016/j.jss.2008.03.066.

A. Appendix

A.1. First Appendix Section

Figure A.1.: A figure

...