Multiagent Learning in an Real Time Strategy Environment

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

English abstract.

Zusammenfassung

Deutsche Zusammenfassung

Contents

List of Figures

List of Tables

1. Motivation

Use Cases:

- Multiagent systems can be used in game theory and financing
- Reconnaissance robots covering a wide area. Communication not always possible.

Aspects:

- Ant-Colony-Optimization, which can be used for learning.
- Emergent Behavior.
- Swarm Intelligence.
- multi-agent reinforcement learning.
- multi-agent learning.
- game theory
- compare these approaches to classical AI approaches in Videogames and RTS's (FSM, B-Trees, GOAP)

1.1. Real-Time Strategy Games for AI Research

Partially Observable Environment (Fog of War), Complex interactions between Units (Micro Dynamic), Concurrent Actions, with multiple Agents on multiple scales (scale of one unit and one game participant).

Torchcraft: (insert bib link here)

- Extremely complex number of combinations (unit states, uncertainty (scouting) and volatile environment states) means that classic planning and search are not practical.
- partial observable environment, hard to quantify performance standard.
- makes a good benchmark for AI.

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`
- `\vskip`

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

Figure 1.1.: SDQ logo

abc	def
ghi	jkl
123	456
789	0AB

Table 1.1.: A table

1.3. Example: Citation

This template is based on `biblatex` and `biber`, which is preferred over the outdated BibTeX software. Please adjust your build environment if necessary (see `https://sdqweb.ipd.kit.edu/wiki/BibTeX-Literaturlisten#biblatex.2Fbiber`)

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.

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

2. Information to sort

2.1. Artificial Intelligence - A modern Approach

2.1.1. 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**.

2.1.2. 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.

2.1.3. 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.

2.1.4. 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.

2.1.5. 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.

2.1.6. Game Theory

p.666

2.1.7. Mechanism Design for Multiple Agents

p.679

2.1.8. Adversarial Search

p.182

2.1.9. Probabilistic Reasoning over Time

p.587

2.1.10. Reinforcement Learning

p.830

2.1.11. Planning Uncertain Movements (Potential Fields)

p.993

2.2. Ant Colony Optimization

2.2.1. 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).

2.3. 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

2.4. References/Papers

$AComprehensiveSurveyofMultiagentReinforcementLearning_{2}008$

$AntColonyOptimizationforlearningBayesiannetwork$

$DistributedArtificialIntelligence_{2}020$

$DistributedArtificialIntelligence_{2}020$

$DistributedCooperativeControlandCommunicationforMulti-agentSystems_{2}021$

$MachineLearningandGames_{2}006$

$PRIMA2020PrinciplesandPracticeofMulti-AgentSystems_{2}021$

$Torchcraft-ALibraryforMachineLearningResearchonReal-TimeStrategyGames_{2}016$

$TheMultiagentPlanningProblem_{2}016$

$SwarmIntelligence_{2}010$

$SwarmIntelligence_{2}012$

$SwarmIntelligence_{2}014$

$SwarmIntelligence_{2}016$

$SwarmIntelligence_{2}018$

$SwarmIntelligence_{2}020$

$ImitativeLearningforRTS_{2}012$

$CombiningStrategicLearningandTacticalSearchinRTSGames_{2}017$

$DeepRTSAGameEnvironmentforDeepReinforcementLearninginReal{-}TimeStrategyGames_{2}018$

$ArtificialIntelligenceTechniquesonReal-timeStrategyGames_{2}018$

$UsingMulti-AgentPotentialFieldsinReal-TimeStrategyGames_{2}008$

$EvaluatingtheEffectivenessofMulti{-}AgentOrganisationalParadigmsinaReal{-}TimeStrategyEnvironme$

$TheStarCraftMulti-AgentChallenge_{2}019$

$Neuroevolutionbasedmulti{-}agentsystemformicromanagementinreal{-}timestrategygames_{2}012$

$AReviewofReal-TimeStrategyGameAI_{2}014$

$DealingwithfogofwarinaRealTimeStrategygameenvironment_{2}008$

3. Fundamentals

Topics:

- multiagent/multibody Systems (MAS).
  - MAS in General
  - MAS Reinforcement Learning
  - MAS Learning for micromanagement in RTS.
  - MAS Movement Problems (Potential Fields).
  - MAS for RTS
- Ant-Colony-Optimization (ACO)
  - ACO for learning.
- Classic RTS AI Approaches.
  - FSM
  - Behavior Trees
  - GOAP

4. Problem and Approaches

Agent:

- performance measure: which one?
- needs information gathering
- uses percepts to find correct action from prior knowledge
- communication is important
- learning while a game is running?

Multiagent:

- multiagent (decentralized planning with coordination) or multibody (centralized planning)?
- multieffector (units can walk and attack) (can be made easier).

Environment:

- partially observable (fog of war).
- cooperative multiagent environment for a given player.
- competitive multiagent environment between players.
- usually deterministic, games may use RNG than it would stochastic.
- stochastic because of partially observability.
- sequentiell (current actions depend on previous actions).
- static environment (when AI is framelocked), otherwise dynamic
- pseudo-continious environment
- known, but partially observable.
- basically hardest case: partially observable, multiagent, stochastic, sequentiell, dynamic, continuous and unknown.

5. Project

- use a video game environment to simulate ants and learning with multi-agent systems.
- could create a survival scenario (vs. nature) and an adversarial scenario (vs. another AI or player).
- create abstracted mechanics that create a learning environment in the context of a video game.

6. Related Works

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

…