Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Projektarbeit (Informatik)

Reinforcement Learning mit einem Multi-Agenten System für die Planung von Zügen

Autoren	Dano Roost Ralph Meier
Hauptbetreuung	Andreas Weiler
Nebenbetreuung	Thilo Stadelmann
Datum	18.09.2019

Zusammenfassung

Zusammenfassung in Deutsch

Abstract

Abstract in English

(Deutschsprachiges Management Summary)

(Englischsprachiges Management Summary)

Vorwort

Stellt den persönlichen Bezug zur Arbeit dar und spricht Dank aus.

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum: Unterschriften:

……………………………………………… ………………………………………………………………

………………………………………………………………..

………………………………………………………………

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Abstract bzw. dem Management Summary mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Inhaltsverzeichnis

1. Einleitung

1.1. Ausgangslage

- Nennt bestehende Arbeiten/Literatur zum Thema -> Literaturrecherche
- Stand der Technik: Bisherige Lösungen des Problems und deren Grenzen
- (Nennt kurz den Industriepartner und/oder weitere Kooperationspartner und dessen/deren Interesse am Thema Fragestellung)

1.2. Zielsetzung / Aufgabenstellung / Anforderungen

- Formuliert das Ziel der Arbeit
- Verweist auf die offizielle Aufgabenstellung des/der Dozierenden im Anhang
- (Pflichtenheft, Spezifikation)
- (Spezifiziert die Anforderungen an das Resultat der Arbeit)
- (Übersicht über die Arbeit: stellt die folgenden Teile der Arbeit kurz vor)
- (Angaben zum Zielpublikum: nennt das für die Arbeit vorausgesetzte Wissen)
- (Terminologie: Definiert die in der Arbeit verwendeten Begriffe)

2. Technical Foundation

2.1. Reinforcement Learning

Basic Definitions

In recent years, major progress has been achieved in the field of reinforcement learning (RL) [1],[2], [3]. In RL, an agent $\mathcal{A}$ learns to perform a task by interacting with an environment $\mathcal{E}$. On every timestep t the agent needs to take an action u. The selection of this action u is based on the current observation s. The success of the agent is measured by reward $\mathcal{R}$ received. If the agent does well, it receives positive reward from the environment, if it does something bad, there is no or negative reward. The goal of the agent $\mathcal{A}$ is now to take an action that maximizes the expected future reward $\mathbb{E}[\mathcal{R}_{t+1} + \mathcal{R}_{t+1} + \mathcal{R}_{t+1} + ...|s_t]$ given the current observation s.
The current observation s_t, also known as the current state is used to determine which action u to take next. An agent can observe its environment either fully or partially.

Value Based vs. Policy Gradient Based Methods

Reinforcement learning methods are categorized into value-based methods and policy-based methods[4],[5]. Those variants differ on how they select an action u from a state s. Value-based RL algorithms work by learning a value function $\mathcal{V}(s)$ through repeated rollouts of the environment. $\mathcal{V}(s)$ aims to estimate the future expected reward for any given state s as precisely as possible. Using this approximation $\mathcal{V}(s)$ we can now select the action u that takes the agent into the next state s_{t+1} with the highest expected future reward. This estimation $\mathcal{V}(s)$ is achieved by either a lookup table for all possible states or a function approximator. In this work, we solely focus on the case that $\mathcal{V}(s)$ is implemented in form of a neural network as function approximator.
The second category of reinfocement learning algrithms are the so called policy gradient based methods. These methods aim to aquire a stochastic policy π that maximizes the expected future reward $\mathcal{R}$ by taking actions with certain probabilities. Taking actions based on probabilities solves an important issue of value based methods, which is, that by taking greedy actions with respect to state s, the agent might not explore the whole state space and misses out on better ways to act in the environment.

Asynchronous advantage actor critic algorithm

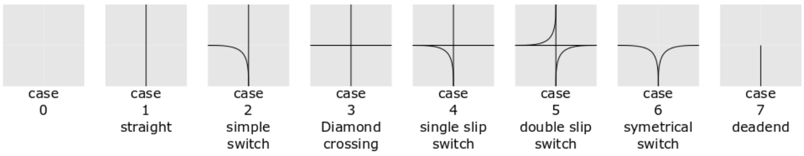
The progress in RL has led to algorithms that combine value based and policy gradient based methods. To enhance the process of learning policy π, the policy loss gets multiplied by the difference between actually received reward $\mathcal{R}$ and the estimated future reward $\mathcal{V}(s)$. TODO: Extend

Relation to this Work

The goal of this work is to apply an RL algorithm to the vehicle rescheduling problem. Based on the work of S. Hubacher (source!!!), we use a distributed RL algorithm that learns a policy to control the traffic of trains on a rail grid. To do so, we use the asynchronous advantage actor critic algorithm [6] and expand its definiton to the use case of multiple agents, similar to [7].

2.2. The Flatland Rail Environment

The flatland environment is a virtual simulation environment provided by the Swiss Federal Railway SBB and the crowdsourcing platform AICrowd. The goal of this environment is to act as a simplified simulation of real train traffic. Using flatland, we can train RL algrithms to control the actions of trains, based on observations on the grid. Flatland has a discrete structure in both its positions and its timesteps.The whole rail grid is composed out of squares that can have connections to neighbouring squares. In certain squares, the rails splits into two rails. On those switches, the agent has to make a decision which action it wants to take. Dependent on the type of switch, there are different actions available.



All rail parts, independent of if it is a switch also allow to take the actions to do nothing (remain halted, or keep driving), to go forward or to brake. The action space is therefore defined by:

$$U = \{\text{Do nothing, go left, go forward, go right, brake}\}$$

It is important to note that trains do not have the ability to go backwards and therefore need to plan ahead to avoid getting stuck. To learn which actions to take, the agents have to learn to adapt to an unknown environment due to the fact that the environments are randomly generated and differ on each episode. Depending on the given parameters, the size and complexity of the grid can be adjusted. This allows for dynamically changing the difficulty for the agents.

The goal of each agent is to reach an assigned target train station as fast as possible. Agents that reach this destination are removed from the grid which means, they can no longer obstruct the path of other trains.

Agent Evaluation

AICrowd an SBB provide a system for agent evaluation. This system evaluates the policy on a number of unknown environments and outputs the percentage of agents that reached their destination as well as the received reward while doing so. The evaluation reward scheme is thereby as follows:

$$R_t = \begin{cases} -1, & \text{if } s_t \text{ is not terminal} \\ 10, & \text{otherwise} \end{cases}$$

All submissions to the flatland challenge are getting graded by the percentage of agents that made it to destionation. (Source) Additionally we use our own evaluation parcour with an increasing difficulty of environments to get more insight into the agents strenghts and weaknesses.

Observations

The flatland environment allows to create observation builders to observe the environment for each agent. While it is possible to observe the whole grid, this does usually not make sense due to the fact that many parts of the rail grid are not relevant to a single train. Flatland offers by default two different observation builders.

GlobalObsForRailEnv creates three arrays with the dimensions of the rectangular rail grid. The first array contains the transition information of the rail grid. For each cells, there are 16 bit values, 4 for each possible direction a train is facing.

TreeObsForRailEnv creates a graph with sections of the grid as nodes from the perspective of the train. This means, only the switches which the train is actually able to take define a single node. As an example, a train on a *case 0* switch heading from north to south is not able to make a decision on this switch and therefore, the TreeObservation does not put the sections before and after the switch into two different nodes but just into a single node.
IMAGE mapping TreeObservations
The nodes of the tree observation offer a number of fields that allow to select specific features to create numeric input vectors for function approximator such as neural networks. The tree observation builder offers 14 distinct features for each rail section. This includes:

- Dist. own target encountered: Cell distance to the own target railway station. Inf. if target railway station for agent is not in this section.
- Dist. other agent encountered: Cell distance to the next other agent on this section.
- Dist. to next branch: The length of this section.
- Dist. min to target: The cell distance to the target after this section is finished.
- Child nodes: The nodes the agent is able to take after this section ends. Each child node is associated with a direction (left, forward, right).

3. Approach and methodology

3.1. Basic considerations

As described under (tODO: ref to first mention), our work is based on the work of S. Huschauer (REF). We take his idea of using the A3C algorithm to solve the flatland problem and try various modifications in an attempt to improve its performance. We proceed by giving an intution, what we want to achieve by changing the specified part, followed by an experiment to either prove or disprove our hypothesis. For training purposes, we started by reimplementing the algorithm by ourselfes. This enabled us from the beginning to gain a deeper understanding of how the algorithm works and where we could find possible areas for improvement. From there, we iteratively added these potential improvements to later compare them against the original version. In this work, we proceed by comparing the final version to versions without these features. It is important to note, that the training process of reinforcement learning and especially multi agent reinforcement learning is hard to evaluate. Depending on the initial weights of the neural networks and the shape of the environments, the performance may vary on each restart. Also, the number of workers can

3.2. A3C implementation for flatland

Originally, the asynchronous advantage actor critic algorithm (A3C) has been designed for use in a single agent environment. By applying it in a multi agent environment, we implicitly convert the environment into a non-stationary environment. While applying A3C in a multi agent setting, the other agents can be viewed as part of the environment. This means, the behaviour of the environment changes while training, due to the fact that the behaviour of the other agents changes.

Gupta et al. [8] finds, that methods like Deep-Q networks (DQN) and Trust region policy optimization (TRPO) are not performing well in a multi agent environment, due to the combination of experience replay and non-stationarity of the environment. We therefore suggest, that it is not recommendable to keep an experience replay buffer with older episodes. Otherwise the sampled experience might represent old agent behaviour which is then learned.

3.3. Enhanced observations

Für die vorliegende Arbeit wurden die unten aufgeführten Programme eingesetzt.

3.4. Distrubuted architecture and parallelism

Für die vorliegende Arbeit wurden die unten aufgeführten Programme eingesetzt.

3.5. Action space reduction

Für die vorliegende Arbeit wurden die unten aufgeführten Programme eingesetzt.

3.6. Entropy balancing

Für die vorliegende Arbeit wurden die unten aufgeführten Programme eingesetzt.

3.7. Agent communication

Für die vorliegende Arbeit wurden die unten aufgeführten Programme eingesetzt.

- (Beschreibt die Grundüberlegungen der realisierten Lösung (Konstruktion/Entwurf) und die Realisierung als Simulation, als Prototyp oder als Software-Komponente)
- (Definiert Messgrössen, beschreibt Mess- oder Versuchsaufbau, beschreibt und dokumentiert Durchführung der Messungen/Versuche)
- (Experimente)
- (Lösungsweg)
- (Modell)
- (Tests und Validierung)
- (Theoretische Herleitung der Lösung)

3.8. (Verwendete Software)

Für die vorliegende Arbeit wurden die unten aufgeführten Programme eingesetzt.

Arbeitsumgebung

- Microsoft Windows 8 developer preview

Virtual Machine

- Oracle VM VirtualBox, Version 3.2.10

CAD Catia

- CATIA, Version 5.19 (in VirtualBox)

Dokumentation

- proTeXt mit TexMakerX 2.1 (SVN 1774), latex-project.org
- Microsoft Visio 2007
- Adobe Acrobat 8 Professional 8.1.6

4. Resultate

- (Zusammenfassung der Resultate)

5. Diskussion und Ausblick

- Bespricht die erzielten Ergebnisse bezüglich ihrer Erwartbarkeit, Aussagekraft und Relevanz
- Interpretation und Validierung der Resultate
- Rückblick auf Aufgabenstellung, erreicht bzw. nicht erreicht
- Legt dar, wie an die Resultate (konkret vom Industriepartner oder weiteren Forschungsarbeiten; allgemein) angeschlossen werden kann; legt dar, welche Chancen die Resultate bieten

6. Verzeichnisse

Literaturverzeichnis

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013. [Online]. Available: https://arxiv.org/pdf/1312.5602.pdf

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. [Online]. Available: https://science.sciencemag.org/content/362/6419/1140

[3] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," 2019.

[4] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, Aug 1988. [Online]. Available: https://doi.org/10.1007/BF00115009

[5] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063. [Online]. Available: http://dl.acm.org/citation.cfm?id=3009657.3009806

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[7] G. Bacchiani, D. Molinari, and M. Patander, "Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning," 2019.

[8] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems*, G. Sukthankar and J. A. Rodriguez-Aguilar, Eds. Cham: Springer International Publishing, 2017, pp. 66–83.

Abbildungsverzeichnis

Tabellenverzeichnis

(Glossar)

In diesem Abschnitt werden Abkürzungen und Begriffe kurz erklärt.

Abk	Abkürzung
XY	Ix Ypsilon
YZ	Ypsilon Zet

Listings

A. Anhang

A.1. Projektmanagement

- Offizielle Aufgabenstellung, Projektauftrag
- (Zeitplan)
- (Besprechungsprotokolle oder Journals)

A.2. Weiteres

- CD mit dem vollständigen Bericht als pdf-File inklusive Film- und Fotomaterial
- (Schaltpläne und Ablaufschemata)
- (Spezifikationen u. Datenblätter der verwendeten Messgeräte und/oder Komponenten)
- (Berechnungen, Messwerte, Simulationsresultate)
- (Stoffdaten)
- (Fehlerrechnungen mit Messunsicherheiten)
- (Grafische Darstellungen, Fotos)
- (Datenträger mit weiteren Daten (z.B. Software-Komponenten) inkl. Verzeichnis der auf diesem Datenträger abgelegten Dateien)
- (Softwarecode)