Reinforcement Learning für Gebäuderegelung (RLG)

Teil 1: Erstellung einer Softwareumgebung

Gabriel Eyyi
Institut für angewandte Informationstechnologie
Information Engineering
8401 Winterthur
eyyigab1@students.zhaw.ch

ABSTRACT

Ein Grossteil des globalen Energieverbrauchs wird alleine durch den Gebäudesektor verursacht. Einsparpotenzial liegt vor allem in der Steuerung der Gebäudetechnik und der Optimierung von verschiedenen Energiequellen. Fortgeschrittene Steuerungstechniken, wie Model Predictive Control (MPC), können die Steuerungen optimieren, sind aber aufgrund der zu hohen Entwicklungskosten für alltägliche Bauprojekte nicht rentabel. Aus diesem Grund wird im Projekt *Reinforcement Learning für Gebäudereglung (RLG)*, anhand lernbasierter Verfahren aus dem Bereich Reinforcement Learning (RL) und anhand eines Simulationsmodells, eine optimale Steuerung von unterschiedlichen Heizungsanlagen entwickelt und untersucht. Ziel des ersten Teilprojekts ist die Entwicklung einer Softwareumgebung, um Steuerungen mithilfe von Reinforcement Learning Verfahren sowie von Simulationsmodellen entwickeln und untersuchen zu können. Durch das Lösen einer klassichen Steuerungstechnikaufgabe wurde die Softwareumgebung experimentell validiert. Das Ergebnis dieses Teilprojekts ist eine Python Library, welche es ermöglicht, Simulationsmodelle von Modellica mithilfe von RL Algorithmen zu untersuchen.

CCS Concepts

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Keywords

Reinforcement Learning, Dymola, Softwareumgebung, FMU Co-Simulation

1. EINLEITUNG

Rund ein Drittel der Schweizer CO2-Emissionen, sowie fast 40 % des globalen Energieverbrauchs werden allein durch den Gebäudesektor verursacht[1],[2]. Massnahmen und Lösungsmöglichkeiten, welche den Energieverbrauch im Gebäudesektor reduzieren können, sind von grosser Bedeutung. Neben dem aktiven Beitrag zum Klimaschutz, können die Eigentümer und Eigentümerinnen von den tieferen Kosten profitieren.

Ein grosses Einsparpotential liegt in der Gebäudesteuerung von Heizung, Lüftung und der Klimatechnik (HLK). Grob geschätzt wird in den Industrienationen die Hälfte der konsumierten Energie für HLK genutzt[3]. Zudem verursachen die Inbetriebnahme von Gebäudesteuerungen, insbesondere von Heizungsanlagen, aufgrund der Komplexität und dem Bedarf an Experten, hohe Kosten[4]. Ein weiteres Einsparpotential liegt in der Steuerung und Optimierung von verschiedenen Energiequellen, sogenannten *Hybrid Energy Systems*. Komplementär zu den erneuerbaren Energiequellen enthalten diese Systeme Hilfsquellen, um die produzierte Energie speichern und den kontinuierlichen Energiebedarf decken zu können[5]. Eine allgemeine Lösungsmöglichkeit, um die Gebäudesteuerung zu optimieren und somit die Energiekosten zu reduzieren, kann mithilfe von fortgeschrittenen Steuerungstechniken, wie *Model Predictive Control (MPC)* realisiert werden. Derartige Gebäudesteuerungen sind für die Optimierung auf ein präzises mathematisches Modell des Gebäudes angewiesen[6]. Obwohl der Einsatz von MPC in Gebäudesteuerungen den Energieverbrauch reduzieren kann, sind zurzeit die Entwicklungskosten von MPC für alltägliche Bauprojekte zu hoch[6]. Eine alternative Lösungsmöglichkeit, welche ohne ein mathematisches Modell auskommen würde, ist der Einsatz von intelligenten Verfahren, wie *Reinforcement Learning (RL)*. Diese Methoden können die Parametrisierung der Steuerung anhand von empirischen Messwerten des Gebäudes und einer definierten Kostenfunktion optimieren[7]. Allerdings werden für die experimentelle Untersuchung synthetische Messwerte benötigt, welche ohne ein mathematisches Modell nicht generiert werden können.

Die Firma RINO Electronics AG entwickelt in Zusammenarbeit mit dem Institut für Angewandte Mathematik und Physik der ZHAW (IAMP) in einem KTI Projekt[1] ein solches mathematisches Modell, um die Steuerung von verschiedenen Heizungsanlagen zu optimieren. Dieses Simulationsmodell wird in der objektorientierten Modellierungsspache Modelica[2] modelliert. Um RL Algorithmen für das Optimieren von Gebäudesteuerung zu untersuchen, stellt das IAMP ein vereinfachtes Modell zur Verfügung. Das Endziel des Projekts *Reinforcement Learning für Gebäuderegelung (RLG)* besteht darin, die Steuerung von unterschied-

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

[1] https://www.kti.admin.ch/kti/de/home.html
[2] https://www.modelica.org/

lichen Heizungsanlagen, wie Wärmepumpen, Photovoltaikanlagen oder Eisspeichersysteme eines Mehrfamilienhauses mithilfe von RL Algorithmen zu lernen und zu optimieren. Damit dieses Endziel erreicht werden kann, soll in einem ersten Teilprojekt eine Softwareumgebung entwickelt werden, die es ermöglicht Steuerungen von Simulationsmodellen von Modelica mithilfe von RL Algorithmen zu entwickeln und zu untersuchen. Der Beitrag der vorliegenden Arbeit zum Projekt RLG ist die Entwicklung und Validierung der Softwareumgebung.

Diese Arbeit ist wie folgt organisiert. In Kapitel 2 werden bestehende Lösungsmöglichkeiten für die Optimierung der Gebäudesteuerung diskutiert. Die Grundlagen von Reinforcement Learning, sowie ein typischer Algorithmus für das Optimieren der Steuerungsaufgaben wird in Kapitel 3 vorgestellt. In Kapitel 4 wird die Architektur der Softwareumgebung beschrieben und in Kapitel 5 wird die Softwareumgebung experimentell validiert. In Kapitel 6 werden die Ergebnisse und zukünftige Arbeiten beschrieben. Schliesslich werden im letzten Kapitel die Erkenntnisse diskutiert.

2. RELATATED WORK

Das oberste Ziel jeder Steuerung von Heizungsanlagen ist das Wohlbefinden der Bewohner und die Optimierung der Energieeffizienz[4]. Frühere Arbeiten im Bereich der Gebäudesteuerung haben gezeigt, dass mit fortgeschrittenen Steuerungstechniken wie MPC vielversprechende Resultate erzielt werden können[8]. Für die Optimierung der Steuerung, benötigen MPC Verfahren ein explizites mathematisches Modell, welches die Dynamik der Gebäudesteuerung beschreibt. MPC verwendet das Modell, um zukünftige Zustände, wie die Veränderung der Umgebungstemperatur über einen bestimmten Zeithorizont, vorhersagen zu können[9]. Mit diesen Vorhersagen ist es Möglich ein Optimierungsproblem zu definieren, welches anhand einer bestimmten Kostenfunktion, wie die Energiekosten, gelöst werden kann[10]. Allerdings sind diese Verfahren stark abhängig von der Präzision des mathematischen Modells des Gebäudes[8]. Als Alternative Steuerungstechnik für Gebäudesteuerungen eignen sich lernbasierte Verfahren wie Reinforcement Learning (RL). Diese Steuerungstechniken benötigen kein mathematisches Modell des Gebäudes (Model-Free-Control). Das RL Verfahren bietet die Möglichkeit intelligente Kontroller (Agent) zu erstellen, welche anhand des Verhaltens der unbekannten Umgebung optimiert werden können[11]. Die Grundlagen von RL werden im Kapitel 3 beschrieben. In verschiedenen Arbeiten wurde das RL Verfahren für die Entwicklung und Optimierung von Gebäudesteuerungen untersucht. Im folgenden Abschnitt werden Arbeiten vorgestellt, welche das Potenzial von RL Algorithmen in der Gebäudesteuerung aufzeigen. In der Arbeit von L. Yang et al. wurden RL Algorithmen erfolgreich für die Steuerung von HLK Systemen eingesetzt. Sie konnten zeigen, dass Steuerungen mit RL Algorithmen gegenüber herkömmlichen regelbasierten Steuerungen bessere Performance aufweisen. Beispielsweise konnte bei PVT-Systeme bis zu 11 % mehr Energie produziert werden. Gemäss den Autoren ist der grosse Vorteil von Reinforcement Learing im Bereich der Gebäudesteuerung jener, bei der sich die Steuerung auf die Bedürfnisse und Vorlieben der Bewohner anpassen lässt[12]. C. Guan et al. untersuchten einen RL Algorithmus für das Optimieren der Steuerung von Energiespeichersystemen[13]. Mithilfe von RL Methoden können Steuerungsprobleme im Bereich der Energiespeicherung effizient gelöst werden. Derartige Probleme können als Markov Prozesse betrachtet werden, da zukünftige Zustände nur abhängig sind vom aktuellen Zustand und von der ausgeführten Aktion. Dabei entspricht der Energiespeicherlevel dem Zustand des Systems und eine Aktion entspricht dem Laden oder Entladen des Energiespeichers. Das Ziel der Steuerung von Energiespeichersysteme war die Minimierung der Kosten des Stromverbrauchs über einen ganzen Tag. In ihrer Arbeit verwendeten sie den $TD(\lambda)$-Learning Algorithmus[14] und konnten die monatlichen Ersparnisse der Energiekosten um 59.8 % verbessern[13].

Diese Arbeiten haben gezeigt, dass RL Algorithmen erfolgreich für die Optimierung in Gebäudesteuerungen eingesetzt werden können. Vor allem haben sie uns ermutigt, RL Algorithmen für die Steuerung von Heizungsanlagen genauer zu untersuchen.

3. GRUNDLAGEN

3.1 Reinforcement Learning

Reinforcement Learning ist eine rechentechnische Methode für die Automatisierung von zielgerichtetem Lernen sowie Entscheidungsfindungen. Es ist neben dem Supervised Learning und Unsupervised Learning ein Teilgebiet von Machine Learning und unterscheidet sich vor allem beim Lernen[7]. Mithilfe von empirischen Daten wird versucht, einen Zustand oder eine Situation einer *Aktion* zuzuordnen, sodass eine numerische Belohnung, der *Reward*, maximiert wird. Dabei interagiert der *Agent* (Kontroller) direkt mit der *Umgebung* und versucht mithilfe von Trial-and-Error-Suche eine optimale Zuordnung eines Zustands zu einer *Aktion* zu finden. Diese Zuordnung wird als *Policy* bezeichnet. Im Gegensatz zu Supervised Learning oder Unsupervised Learning stehen dem Agenten weder exemplarische Beobachtungen, noch die genaue Dynamik der Umgebung zur Verfügung. Stattdessen muss der Agent verschiedene Aktionen ausprobieren und die erhaltenen Auswirkungen untersuchen. Diese Auswirkungen können unmittelbar oder verzögert wahrgenommen werden und geben an, wie zielführend die getätigte Aktion war[7]. Im Vergleich zu den anderen Teilgebieten von Machine Learning eignet sich Reinforcement Learning Algorithmen, um Steuerungsprobleme (Control Tasks) von Systemen zu lösen, bei der die Dynamik des Systems unbekannt ist[15].

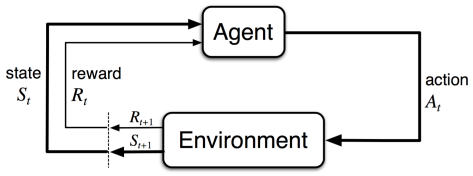

Abb. 1: Schematische Darstellung des Reinforcement Learning Problems[7].

In Abbildung 1 ist das allgemeine Reinforcement Learning Problem schematisch dargestellt. Zu jedem Zeitpunkt t führt der Agent, aufgrund des aktuellen Zustandes S_t, eine Aktion A_t aus. Dabei wird der Zustand S_t von der Umgebung bereitgestellt. Die Aktion A_t bewirkt eine Änderung des Zustandes in S_{t+1}. Nach der Überführung in den nächsten Zustand erhält der Agent jeweils den Reward R_t. Im

Allgemeinen interagiert der Agent mit der Umgebung, um sein langfristiges Ziel zu erreichen. Wobei das langfristige Ziel des Agenten ein Mass für die kumulierten Rewards über eine endliche oder unendliche Entscheidungssequenz ist[16].

Q-Learning.

Q-Learning zählt zu den bedeutendsten Algorithmen in Reinforcement Learning und wurde von Waktins und Dayan[17] entwickelt. Q-Learnig ist ein off-policy Algorithmus und gehört zu der Familie der Temporal-difference (TD) Learning Algorithmen, welche imstande sind, ein optimales Verhalten ohne ein vollständiges Modell der Umgebung zu erlernen (Modellfreies Lernen)[18]. Off-policy Algorithmen sind in der Lage unabhängig von ihrer Policy, die Action-Value Funktion Q zu approximieren, um die optimale Funktion $Q^\star$ zu erhalten [7],[17]. Der Algorithmus Q-Learning ist durch folgende Gleichung definiert:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \underbrace{[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]}_{TD-Fehler\Delta}, \quad (1)$$

Wobei eine Aktion $a \in A(s)$ ist und $A(s)$ die Menge aller möglichen Aktionen des Zustandes s beschreibt. Die Learningrate $\alpha \in (0, 1)$ bestimmt den Einfluss der neu gewonnen Informationen (Δ) und der Diskontierungsfaktor $\gamma \in [0, 1]$ bestimmt mit welchem Ausmass aktuelle Entscheidungen zukünftige Rewards beeinflussen sollen[7].

Exploration vs Exploitation Trade-off.

In modellfreien Lernverfahren ist der Reward des Agenten abhängig vom gelernten Wissen über die Umgebung. Aus diesem Grund benötigen derartige Lernverfahren eine stochastische Komponente in der Aktionsauswahl, die sicherstellt, dass möglichst viele und präzise Informationen über die Umgebung gelernt werden können. Der Agent verfügt über zwei Möglichkeiten Informationen über die Umgebung zu erlernen. Er kann die Umgebung erkunden (Exploration) oder vorhandenes Wissen über die Umgebung ausnutzen (Exploitation). Da die Ausnutzung des vorhanden Wissens immer zu einer minimalen Verbesserung führt, jedoch die Erkundung auch zu einer Verschlechterung führen kann, muss der Agent die Kosten und Nutzen dieser zwei Möglichkeiten abwägen (Trade-off). Mittels einer einfachen Suchstrategie, wie der *ϵ-greedy* Methode, kann eine Balance zwischen Exploration und Exploitation gefunden werden. Bei der *ϵ-greedy* Methode ist ϵ die Explorationsrate und beschreibt die Wahrscheinlichkeit eine zufällige Aktion zu wählen, um die Umgebung zu erkunden. Wobei $1 - \epsilon$ die Wahrscheinlichkeit beschreibt, vorhandenes Wissen auszunutzen. Dabei wird die Aktion so gewählt, dass der grösste Reward erreicht werden kann[19], [7], [20].

4. SOFTWAREUMGEBUNG

Das Ziel dieses Teilprojekts ist die Entwicklung einer Softwareumgebung, um Steuerungen von Simulationsmodellen mithilfe von RL Algorithmen zu entwickeln und zu untersuchen. Die Softwareumgebung besteht grundsätzlich aus den folgenden drei Komponenten:

- Komponente 1: Eine Python Schnittstelle für das externe Steuern von Simulationen.
- Komponente 2: Reinforcement Learning Library für das Anwenden und Vergleichen von verschiedenen Algorithmen.
- Komponente 3: Koppelung der ersten beiden Komponenten.

Komponente 1: Steuerung der Simulation.

Die Ausgangslage für die Entwicklung der ersten Komponente sind Modelica[3] Simulationsmodelle. Modelica ist eine nicht proprietäre, objektorientierte und gleichungsbasierte Modellierungssprache für komplexe physikalische Modelle. Es existieren verschiedene Implementierungen und grafische Entwicklungsumgebungen (IDE) für Modelica. Für dieses Projekt wurde die IDE Dymola 2017[4] verwendet. Um Simulationsmodelle von extern starten zu können, wurde das Functional Mock-up Interfaces (FMI) für Co-Simulation verwendet. FMI ist ein Standard für die einheitliche Kommunikation und Austausch von Modellen zwischen verschiedenen Modellierungstools und Simmulationstools[21]. Entwicklungsumgebungen, die den Standard unterstützen, können die entwickelten Simulationsmodelle als komprimierte Archive (Functional Mock-up Units FMUs) exportieren und untereinander austauschen. Diese Archive enthalten einen eigenen numerischen Solver. Im Allgemeinen liefert eine Co-Simulation für einen Input u_n und einer Schrittweite τ, sowie allfällige Parameter p, den Output y_{n+1} zu der Zeit $T_n + \tau = T_{n+1}$

$$y_{n+1} = \Phi(\tau, u_n; p). \quad (2)$$

Dabei werden die internen Zustände automatisch nachgeführt[21]. Dies erlaubt die Verwaltung und Steuerung von Simulationen durch ein externes Tool[22].

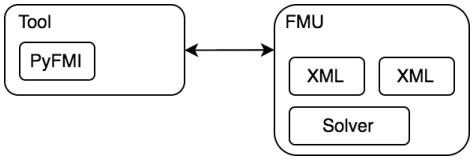

Abb. 2: Darstellung einer Co-Simulations FMU und die Verbindung zu dem Simulationstool vgl. [21].

Komponente 2: Reinforcement Learning Library.

Die zweite Komponente wurde mithilfe des Reinforcement Learning Toolkit OpenAI Gym[23] realisiert. Das Toolkit ist eine Python Framework für RL Probleme. Das Python Toolkit OpenAI Gym entstand aus der OpenAI Gemeinschaft, welches sich zum Ziel gesetzt hat, sichere Artificial Intelligence (AI) zu entwerfen und die Vorteile von AI gleichberechtigt und gleichmässig zu verteilen[23]. Das Framework gibt die Architektur vor, um RL Algorithmen auf verschiedene Umgebungen anwenden zu können. Die Architektur des Frameworks basiert auf den zwei Grundbausteinen Agent und Environment von RL, welche in Abbildung 1 dargestellt sind.

[3] https://www.modelica.org
[4] http://www.cenit.com

Komponente 3: Koppelung.
Die Koppelung der beiden ersten Komponenten wurde mithilfe des Python Moduls PyFMI[5] umgesetzt. Das Modul bildet die FMI Schnittstelle für Python ab und bietet zusätzlich noch weitere Funktionen an, um Experimente zu starten oder komplexe dynamische Systemmodelle zu evaluieren. Um RL Algorithmen auf Simulationen von Dymola anwenden zu können, wurde ein neues Environment *DymolaEnv* in OpenAI Gym entwickelt, welches das Interface gym.core implementiert. Mithilfe der FMUs kann für ein Modellica Model der nächste Zustand simuliert werden.

Diese drei Komponenten bieten nun die Möglichkeit Simulationsmodelle von Dymola mit RL Algorithmen zu untersuchen. Allerdings müssen zunächst die Zustände und Aktionen des Steuerungsproblems klar definiert werden.

5. VALIDIERUNG

Die Softwareumgebung wurde mithilfe des klassischen Steuerungsproblem *Cart Pole Balancing Task* (Inverses Pendel) validiert. Die Steuerungsaufgabe eignet sich aufgrund des kleinen diskreten Aktions- und Zustandsraums, sowie der zahlreichen Lösungen gut für die Validierung der Softwareumgebung.

5.1 Cart Pole Balancing Task

Das Ziel dieser Kontrollaufgabe besteht darin, einen Pendel (Stage), welches mit einem Scharnier am Wagen befestigt ist, zu balancieren. Die Abbildung 3 stellt das Problem grafische dar. Für das Balancieren des Pendels kann der Wagen über eine horizontalen Kraft $\vec{F}$ gesteuert werden. Ein Misserfolg findet statt, wenn die Stage 12 ° von der Senkrechten (Lot) abweicht oder der Wagen den Rand berührt. Diese Aufgabe kann episodisch behandelt werden, wobei die Episoden jeweils die wiederholten Versuche sind, um die Stage zu balancieren. Der Reward beträgt +1 für jeden Zeitschritt, bei der die Stage balanciert werden konnte, ansonsten 0. Damit entspricht der Output die Anzahl Zeitschritte bis zum Misserfolg. Das Inverse Pendel wurde gleich parametrisiert wie im Beispiel von R. Sutton[7], [23]. Die Anfangsbedingungen und Anfangszustände sind in Tabelle 1 und Tabelle 2 aufgelistet.

Paramter	Wert
M	1 kg
m	1 kg
l	0.5 m
τ	0.02 s
θ_0	90 °
$x_{threshold}$	±2.4 m

Tabelle 1: Parameter des Cart Pole Modells.

Das Balancieren des Pendels wurde mithilfe der Simulation des Modells und des Q-Learning Algorithmus gelernt. Die Simulation liefert für eine bestimmte Aktion den nächsten Zustand. Der Zustand S ist definiert als Quadtrupel (s, v, θ, ω), wobei s die Position des Wagens, v die Geschwindigkeit des Wagens, θ der Winkel des Pendels bezüglich des Lots und ω die Winkelgeschwindigkeit des Pendels beschreibt. Die kontinuierliche Zustandsräume, welche in Tabelle 2 aufgelistet sind wurden zunächst diskretisiert.

[5]https://pypi.python.org/pypi/PyFMI

Menge	Werte
Zustand x	$S = \{x \mid -2.4 \leq x \leq 2.4\}$
Zustand v	$S = \{v \mid -1 \leq x \leq 1\ \}$
Zustand θ	$S = \{\theta \mid -2 \leq x \leq 2\ \}$
Zustand ω	$S = \{\omega \mid -3.5 \leq x \leq 3.5\ \}$
Aktion A	$A = \{-1, 1\}$
Reward R	$R = \{0, 1\}$

Tabelle 2: Aktions- und Zustandsräume.

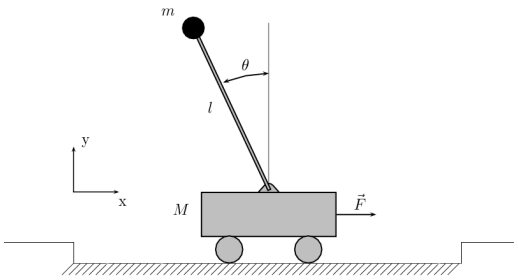

Abb. 3: Darstellung des Cart Pole Balancing Tasks[24].

Die Ausgangslage ist in Abbildung 3 grafisch dargestellt.

5.2 Resultat

Die Kontrollaufgabe wurde auf zwei verschiedenen Environments getestet. Als Algorithmus wurde eine Q-Learning Implementierung von C. Aguayo[25] verwendet. Das Resultat ist in Abbildung 4 dargestellt. Es wurden zwei Experimente durchgeführt. Im ersten Experiment wurde der Algorithmus auf dem *Classic Control Environment (CartPole* von OpenAI angewendet (grüne Kurve) und beim zweiten Experiment wurde der Algorithmus auf dem neu entwickelten *DymolaEnv (DymolaInvertedPendulum)* angewendet (blaue Kurve). In der Abbildung 4 ist ersichtlich, dass die Länge der Episoden pro Episode zunimmt, dies bedeutet, das der Algorithmus bei beiden Environments lernt. Bei der grünen Kurve wird nach 22 Episoden und bei der blauen Kurve nach ca. 155 Episoden die maximale Episodenanzahl erreicht. Zusätzlich ist ersichtlich, dass die Streuung der Episodenlänge bei der grünen Kurve nach 155 Episoden grösser ist als bei der blauen Kurve.

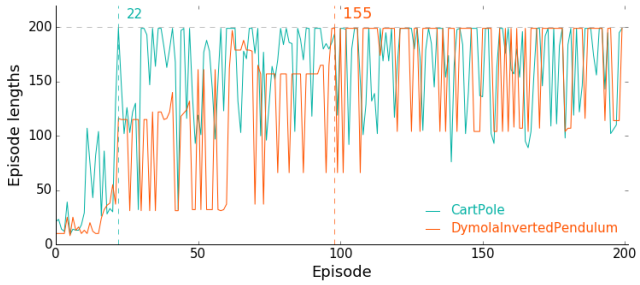

Abb. 4: Entwicklung der Episodenlänge der beiden Experimente

6. ERGEBNISSE UND FUTURE WORK

Das Ergebnis des ersten Teilprojekts von RLG ist eine Softwareumgebung, welche ermöglicht, einfach und schnell Steuerungen von dynamischen Systemen, die in Modelica modelliert wurden, mithilfe von RL Algorithmen zu untersuchen, mit dem Ziel eine optimale Steuerung zu entwicklen. Die Softwareumgebung bietet die Möglichkeit verschiedene RL Algorithmen auf das gleiche Steuerungsproblem zu testen. Damit das Endziel des Projekts RLB, RL Algorithmen für Gebäudesteuerung, entwickelt werden kann, müssen zunächst RL Algorithmen für Steuerungsaufgaben mit grösseren kontinuierlichen Aktions- und Zustandsräume untersucht werden. Die Diskretisierung der Aktions- und Zustandsräume bei solchen Steuerungsaufgaben ist nicht empfehlenswert, da aufgrund der feinen Abstimmung die Anzahl der möglichen Kombinationen explodiert und wichtige Informationen über die Struktur des Problems verloren gehen können[26], [27]. Die Untersuchung solcher Zustandsräume ist schwierig und kann nicht effizient durchgeführt werden[26]. Eine Möglichkeit könnte der Einsatz von Methoden sein, welche die Q-Funkion approximieren[28] oder der Einsatz von *Deep Reinforcement Learning*[26] Algorithmen. Diese Algorithmen bieten die Möglichkeit Policies in hochdimensionalen kontinuierliche Aktionsräume zu lernen[26]. Der nächste Schritt in diesem Projekt wäre somit die Untersuchung von derartigen Algorithmen.

7. ZUSAMMENFASSUNG & DISKUSSION

In dieser Arbeit konnte erfolgreich eine Softwareumgebung bereitgestellt werden, um Simulationsmodelle von Dymola bzw. von Modelica für Reinforcement Learning Algorithmen zu verwenden. Die Softwareumgebung baut auf dem OpenAI Toolkit Gym auf und lässt sich erweitern. Die Validierung der Softwareumgebung wurde mithilfe einer klassischen Steuerungsaufgabe durchgeführt und die Resultate mit anderen Umgebungen verglichen. Die Softwareumgebung bietet nun die Möglichkeit komplexe Gebäudesteuerungen, welche in Modelica modelliert wurden, mithilfe von Reinforcement Learning Algorithmen zu untersuchen. Mit diesem Teilprojekt wurde die Grundlage geschaffen, um RL Algorithmen für die Steuerung von verschiedenen Heizungsanlagen zu untersuchen. Wobei für die Untersuchung und das Lösen derartiger Steuerungsprobleme, die Definition des Aktions- und Zustandsraums entscheidend ist[29].

8. DANKSAGUNG

Der Autor bedankt sich bei Thilo Stadelmann für die hilfreichen Diskussionen und bei Christian Jäger, Peter Bolt und Rudolf Füchsling für die Unterstützung und die zur Verfügung gestellten Simulationsmodelle.

9. REFERENCES

[1] Bundesamt für Umwelt BAFU. Klimapolitik: Gebäude. http://www.bafu.admin.ch/klima/13877/14510/14513/index.html?lang=de, 2016. Accessed: July 2016.

[2] Jens Laustsen. Energy efficiency requirements in building codes, energy efficiency policies for new buildings. *International Energy Agency (IEA)*, pages 477–488, 2008.

[3] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[4] Anastasios I Dounis and Christos Caraiscos. Advanced control systems engineering for energy and comfort management in a building environment—a review. *Renewable and Sustainable Energy Reviews*, 13(6):1246–1261, 2009.

[5] F Bonanno, G Capizzi, A Gagliano, and C Napoli. Optimal management of various renewable energy sources by a new forecasting method. In *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pages 934–940. IEEE, 2012.

[6] David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S Smith. Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis. *IEEE Transactions on Control Systems Technology*, 24(1):1–12, 2016.

[7] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[8] Frauke Oldewurtel, Alessandra Parisio, Colin N Jones, Dimitrios Gyalistras, Markus Gwerder, Vanessa Stauch, Beat Lehmann, and Manfred Morari. Use of model predictive control and weather forecasts for energy efficient building climate control. *Energy and Buildings*, 45:15–27, 2012.

[9] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[10] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999.

[11] Hitesh Shah and M Gopal. Model-free predictive control of nonlinear processes based on reinforcement learning. *IFAC-PapersOnLine*, 49(1):89–94, 2016.

[12] Lei Yang, Zoltan Nagy, Philippe Goffin, and Arno Schlueter. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 156:577–586, 2015.

[13] Chenxiao Guan, Yanzhi Wang, Xue Lin, Shahin Nazarian, and Massoud Pedram. Reinforcement learning-based control of residential energy storage systems for electric bill minimization. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 637–642. IEEE, 2015.

[14] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[15] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[16] Simeng Liu and Gregor P Henze. Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis. *Energy and buildings*, 38(2):148–161, 2006.

[17] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[18] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial*

intelligence: a modern approach, volume 2. Prentice hall Upper Saddle River, 2003.

[19] Marco Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, Learning, and Optimization*, 12, 2012.

[20] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[21] Christian Andersson. *Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface*. PhD thesis, Lund University, 2016.

[22] Jens Bastian, Christop Clauß, Susann Wolf, and Peter Schneider. Master for co-simulation using fmi. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 63, pages 115–120. Linköping University Electronic Press, 2011.

[23] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[24] Krishnavedala Wikimedia Commons. Cart-pendulum. https://commons.wikimedia.org/wiki/File:Cart-pendulum.svg, 2012. Accessed: July 2016.

[25] Carlos Aguayo. Q learning implementation. https://gym.openai.com/algorithms/alg_0eUHoAktRVWWM7ZoDBWQ9w, 2016. Accessed: July 2016.

[26] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[27] Daniel N Nikovski. Fast reinforcement learning in continuous action spaces, 1998.

[28] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In *Australasian Joint Conference on Artificial Intelligence*, pages 417–428. Springer, 1999.

[29] Martin Riedmiller. 10 steps and some tricks to set up neural reinforcement controllers. In *Neural Networks: Tricks of the Trade*, pages 735–757. Springer, 2012.