DISSERTATION FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Tamás Kegyes

University of Pannonia
2025

Reinforcement learning applications for selected industrial optimization problems

Thesis for obtaining a PhD degree in the
Doctoral School of Information Science and Technology of the University of Pannonia

in the branch of Computer Sciences

Written by **Tamás KEGYES**

Supervisors: **Prof. Dr. habil. János ABONYI**
Dr. Zoltán SÜLE

Propose acceptance (yes / no)

……………………………………………….
Prof. Dr. habil. János ABONYI
(supervisor)

Propose acceptance (yes / no)

……………………………………………….
Dr. Zoltán SÜLE
(supervisor)

As reviewer, I propose acceptance of the thesis:

Name of Reviewer: ………………………….……… yes /no

………………………….………
(reviewer)

Name of Reviewer: ………………………….……… yes /no

………………………….………
(reviewer)

The PhD-candidate has achieved ………% at the public discussion.

Veszprém,

………………………….………
(Chairman of the Committee)

The grade of the PhD Diploma ……………………….

Veszprém,

………………………….………
(Chairman of UDHC)

University of Pannonia

Doctoral (PhD) Dissertation

Reinforcement learning applications for selected industrial optimization problems

DOI:10.18136/PE.2025.948

Author:
Tamás Kegyes

Supervisors:
Dr. Zoltán Süle
Prof. Dr. habil. János Abonyi

A dissertation submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Doctoral School of Information Science and Technology
of University of Pannonia

Department of Computer Science and Systems Technology

University of Pannonia
2025

Pannon Egyetem

Doktori (PhD) Értekezés

Megerősítéses tanulás alkalmazásai válogatott ipari optimalizálási feladatokon

Szerző:
Kegyes Tamás

Konzulensek:
Prof. Dr. habil. Abonyi János
Dr. Süle Zoltán

Értekezés doktori (PhD) fokozat elnyerése érdekében
*a **Pannon Egyetem***

Informatikai Tudományok Doktori Iskola
keretében

Rendszer- és Számítástudományi Tanszék

Pannon Egyetem
2025

Science progresses best when observations force us to alter our preconceptions.

Vera Rubin

A tudomány akkor halad előre a leginkább, amikor a megfigyelések arra késztetnek bennünket, hogy megváltoztassuk előfeltevéseinket.

Rubin Vera

UNIVERSITY OF PANNONIA

Abstract

Faculty of Information Technology
Department of Computer Science and Systems Technology

Doctor of Philosophy

Reinforcement learning applications for selected industrial optimization problems

by Tamás KEGYES

Exploiting the potential of the huge amount of data generated by the Industry 4.0 revolution requires more complex optimization methods. Reinforcement Learning (RL) methods give efficient solutions for a wide range of difficult problems in multiple areas, but nowadays it is still under-evaluated in industrial applications. A major advance of using RL methods is that they can serve solutions not only for a single problem but for a whole problem scheme. In this work, I give a comprehensive overview of RL applications in the field of Industry 4.0 and present a questionnaire guideline to identify the best-fitting RL methods based on the problem properties.

As environmental aspects become increasingly important, disassembly problems have become the focus of researchers. Hence, disassembly problems are not the straight inverses of assembly problems, and the conditions are not standard, disassembly optimization solutions require human control and supervision. I present an integrated novel heuristic to target a dynamically pre-filtered action space for the RL agent (dlOptRL algorithm) which significantly raised the efficiency of the learning path.

Finally, I define a new problem class of constrained stochastic graph traversal problems (CSGTP), and prove that only the disassembly line balancing problems, but also shortest path problems and Hamiltonian pathfinding problems belong to that. I present the Q-compression method, which is applicable for solving mixed-integer optimization problems, especially CSGTP as well by using a dynamic discrete representation of the state space. The results contain further potential for delivering highly automated optimization techniques in a wide range of industrial problems.

PANNON EGYETEM

Kivonat

Műszaki Informatikai Kar
Rendszer- és Számítástudományi Tanszék

Philosophiæ Doctor

Megerősítéses tanulás alkalmazásai válogatott ipari optimalizálási feladatokon

írta: Kegyes Tamás

Az Ipar 4.0 forradalom által generált hatalmas adatmennyiségben rejlő lehetőségek kiaknázása összetettebb optimalizálási módszereket igényel. A megerősítéses tanulási (reinforcement learning, RL) módszerek számos területen nyújtanak hatékony megoldást komplex problémák széles körére, de még mindig alulértékeltek az ipari alkalmazásokban. Az RL módszerek alkalmazásának érdemi előnye, hogy nem csak egyetlen problémára, hanem egy egész problémasémára kínálnak megoldást. Jelen dolgozatomban átfogó áttekintést adok az RL alkalmazásokról az Ipar 4.0 területén, és egy kérdőív alapú útmutatót is bemutatok a legmegfelelőbb RL módszerek azonosításához a megoldandó probléma tulajdonságai alapján.

A környezeti szempontok előtérbe kerülésével, a szétszerelési problémák is mindinkább a kutatók érdeklődésének fókuszába kerülnek. Minthogy a szétszerelési problémák nem az összeszerelési problémák puszta inverzei, és a feltételei sem standardak, így azok megoldása emberi irányítást és felügyeletet igényel. A dolgozatban bemutatok egy új integrált heurisztikát, amelyben az RL ágens egy előszűrt akciótérből választja ki a döntését (dlOptRL algoritmus), ezáltal jelentősen növelve a tanulási hatékonyságot.

Végül bevezettem a korlátozott sztochasztikus gráf bejárási problémaosztályt, és igazoltam, hogy nemcsak a szétszerelési problémák, hanem legrövidebb út-, és Hamilton útkeresési problémák is ide tartoznak. Majd bemutatom a Q-tömörítéses RL módszert, ami az állapottér dinamikus diszkrét reprezentációja révén alkalmas kevert egészértékű optimalizálási feladatok megoldására, különös tekintettel a fenti problémaosztály elemeire. Az eredmények további potenciált rejtenek magasan automatizált optimalizálási technikák fejlesztésére ipari problémák széles körében.

PANNONISCHE UNIVERSITÄT

Auszug

Fakultät für Ingenieurwissenschaften
Abteilung für Verfahrenstechnik

Doktor der Philosophie

Anwendungen zum verstärkenden Lernen für ausgewählte industrielle Optimierungsprobleme

von Tamás Kegyes

Um das Potenzial der riesigen Datenmengen auszuschöpfen, die durch die Industrie-4.0-Revolution generiert werden, sind komplexere Optimierungsmethoden erforderlich. Bestärkendes Lernen (Reinforcement-Learning, RL)-Methoden bieten effiziente Lösungen für eine Vielzahl schwieriger Probleme in verschiedenen Bereichen, werden jedoch in industriellen Anwendungen heutzutage noch unterschätzt. Ein großer Vorteil der Verwendung von RL-Methoden besteht darin, dass sie nicht nur Lösungen für einzelne Probleme, sondern für ganze Problemschemata bieten können. In dieser Arbeit gebe ich einen umfassenden Überblick über RL-Anwendungen im Bereich Industrie 4.0 und präsentiere einen Fragebogenleitfaden, um die am besten geeigneten RL-Methoden basierend auf den Problemeigenschaften zu identifizieren.

Da Umweltaspekte zunehmend an Bedeutung gewinnen, rücken Demontageprobleme in den Fokus der Forschung. Demontageprobleme sind nicht einfach die direkten Gegenteile von Montageproblemen, und die Bedingungen sind nicht standardisiert, weshalb Demontageoptimierungslösungen menschliche Kontrolle und Überwachung erfordern. Ich stelle eine integrierte neue Heuristik vor, die darauf abzielt, einen dynamisch vorfiltrierten Aktionsraum für den RL-Agenten (dlOptRL-Algorithmus) anzusprechen, was die Effizienz des Lernpfads erheblich erhöht.

Abschließend definiere ich eine neue Problemklasse von Eingeschränkte stochastische Graph-Traversierungsprobleme (ESGTP) und zeige auf, dass dazu nicht nur Ausbalancieren Probleme von Demontagelinien gehören, sondern auch kürzeste Pfadprobleme und Hamiltonsche Pfadfindungsprobleme. Ich stelle die Q-Kompressionsmethode vor, die geeignet ist, gemischt-ganzzahlige Optimierungsprobleme zu lösen, insbesondere ESGTP, indem sie eine dynamische diskrete Darstellung des Zustandsraums verwendet. Die Ergebnisse bergen weiteres Potenzial für die Bereitstellung hochautomatisierter Optimierungstechniken für eine breite Palette industrieller Probleme.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisors, Prof. Dr. habil. János Abonyi and Dr. Zoltán Süle. Words cannot adequately convey my appreciation for their guidance and support throughout my doctoral studies. They provided invaluable academic mentorship and demonstrated immense patience, even when I made the journey more challenging for all of us. Their exemplary way of life, human values, and behavior within the research community have given me more than any scientific book ever could. I learned a lot from them and they were always very helpful. Many thanks for the opportunity to work together; I will never forget these years!

I also want to extend my heartfelt thanks to my family and friends, whose unconditional love and support sustained me during the countless hours I spent researching and preparing publications. I am fortunate to have their understanding and patience, which were instrumental in enabling me to complete my studies and write my thesis.

Lastly, I would like to take this opportunity to thank my teachers, Ph.D. fellows, colleagues, and all of the professional community around me who offered constant encouragement. Engaging in discussions with them and receiving their support was immensely beneficial throughout my academic journey.

Contents

Chapter 1

Introduction

Reinforcement learning (RL) has a significant opportunity to revolutionize artificial intelligence (AI) applications by serving a novel approach to machine learning (ML) developments that allows the user to handle large-scale problems efficiently. These techniques, together with widely used Internet of Things (IoT) tools, have opened up new possibilities for optimizing complex systems, including domains such as logistics, project planning, scheduling, and other industry-related domains. Extracting this potential can result in fundamental progress in the transformation of Industry 4.0 [1]. Vertical and horizontal integration will be strengthened during this digital transformation, flexibility should be increased, and human control and supervision should be focused [2], [3]. Furthermore, the data produced by the integrated tools is increasing exponentially, which requires a higher level of autonomous processes and decisions. Reinforcement learning is valuable for developing self-optimizing and self-organizing Industry 4.0 solutions. The main challenge of creating these applications is that several methods, techniques, and a wide range of parameters must be defined. As the definition of these parameters requires detailed knowledge of the nature of the RL algorithms, the first main goal of this work is to provide a comprehensive overview of the RL methods from the perspective of Industry 4.0 and smart manufacturing and to conclude tendencies of how to apply RL methods for different problem types.

The increasing environmental pressures caused by human activities drive governments and organizations to limit or reduce their footprints. As an outcome, the European Commission declared this a strategic goal. A new Circular Economy Action Plan for a cleaner and more competitive Europe has been announced to transform the economy and society sustainably. There is a wide range of recommendations and proposals on how these issues should be handled, and the key point is to reduce consumption and increase the recycling rate. The efforts led to the creation of the concept of circular economy, which is more complex than just manufacturing optimization because it also covers the optimization steps of the supply chain, disassembly, and recycling. The circular economy optimization goal covers the recollecting and recycling processes that can be a bit more complex because these are more stochastic and less controlled. The most basic models assumed deterministic inputs, but advanced models started to handle the uncertainty observed in real problems. Among others, the source materials and their distributions are described by stochastic variables. Similarly, the disassembly tasks, their required process times, and the demand values of the recycling steps can also be stochastic attributes in some models.

Supply chain optimization and assembly line balancing have been intensively studied since the 1950s, while the focused analysis of disassembly lines started almost 40 years later. The second goal of my research was to explain in detail how to construct a practical implementation for disassembly line balancing problems and discover the advantages and disadvantages of using RL methods.

From a broader aspect, further similar optimization problems can be found, in which formulations lead to mixed-integer (or mixed continuous-discrete) optimization problems. The third goal of my work was to identify and describe a greater problem class, which led me to constrained stochastic graph traversal problems. Although efficient solutions for the classical shortest pathfinding problems are known, these methods do not work with constraints and stochastically distributed distances, which motivated me to turn to the reinforcement learning method. In contrast with the most commonly used gradient-based solutions of mixed-integer optimization problems, As the fourth goal of my research, I focused on a new discretization approach for reducing the state space into a purely discrete space representation. I applied the iterative policy optimization method to determine the solution.

I identified a diverse set of problems belonging to the constrained graph traversal class. Although the shortest pathfinding problem is a well-known combinatorial optimization problem, its parameters are difficult to define precisely. It is easy to see that the optimal route problem of a truck is practically a constrained shortest pathfinding problem, where the constraints describe the working hours limits of the driver and the availability of parking or the fuel capacity limits. The daily planning of an electric delivery van is a constrained Hamiltonian pathfinding problem, where the constraints are based on the limits of the battery capacity and charging options or the opportunities for battery exchange. Furthermore, disassembling all product components after their life cycle is also a constrained graph traversal problem, where a precedence graph describes component removal dependencies and the constraints limit the use of parallel workstations. Finally, I need to point out that, in real-life problems, these problems are rather stochastic optimization tasks than discrete ones.

Reinforcement learning (RL) can be an obvious solution for sequential decision-making processes such as step-by-step pathfinding. The objective function should be implemented in the reward function and violation of constraints. However, additional difficulties arise when considering stochastic effects or measurement uncertainty: some continuous components must be integrated into the state space, which becomes mixed discrete-continuous. There are successful methods to solve the mixed integer problem using gradient-based or deep RL methods. RL techniques can also solve constrained combinatorial optimization problems. Moreover, stochastic shortest-path problems also have RL solutions. Compared to them, my research aims to discover a new algorithm by performing discretization steps using the DBSCAN clustering method and applying a discrete RL method for mixed continuous-discrete constrained stochastic graph traversal problems.

According to the goals of my research identified above, the structure of my dissertation is as follows. Chapter 2 focuses on an extensive review of RL-based solutions in Industry 4.0 applications and stands for the following major parts.

- First, in Section 2.1, I summarize the motivation for presenting a detailed overview of RL applications in Industry 4.0 solutions and the applied methodology of the literature overview.
- Next, in Section 2.2, I give a short general introduction to the reinforcement learning framework and summarize some of the main mathematical properties behind the RL techniques. In Sections 2.2.1.2-2.2.1.9, I provide a compact overview of 18 different RL methods. Furthermore, I present a classification of RL methods that allows the reader to have a map for further discussion.

- As a next step, in Section 2.3, I summarize the key findings of the systematic literature review, provide a hands-on reference for further research and discuss the conclusions of the general trends.
- Finally, in Section 2.4, I give a detailed guideline to support the reader in following an appropriate methodology to set up an RL solution and help to choose a suitable RL method for the different problems.

Chapter 3 discusses the disassembly line balancing problem, including an improved RL solution, and has the following major elements:

- Section 3.1 Introduction to disassembly line balancing optimization
- First, in Section 3.2, I give a problem formulation of disassembly line optimization problems. I summarize the notations, declare the objective function components, and present an MIQP formulation with fewer decision variables. It enables solving larger-size problems with the widely used solvers while keeping the limitations of the decision variables.
- Section 3.3 presents all the necessary steps to prepare for an RL solution to the previously defined disassembly problem. According to it, in Section 3.3.2, I describe a novel algorithm called DLOPTRL. It contains a built-in heuristic to minimize the applicable action space and speed up the learning of an RL agent. My algorithm highlights how RL methods can be combined with problem-specific heuristics for efficient self-learning solutions. I also point out that my algorithm belongs to the Heuristically Accelerated Reinforcement Learning class.
- Section 3.4 describes two commonly used use cases for which I summarize a MIQP solution as a reference and the results of the DLOPTRL learning path. I show that my solution effectively approximates the optimal solution without prior preparations.

Chapter 4 presents a greater problem class that leads to mixed-integer optimization problems, discusses a new RL method to solve it, and consists of the following key components:

- In Section 4.2, I define the class of constrained stochastic graph traversal problems and formulate the constrained shortest pathfinding problem

(CSPP), the constrained Hamiltonian pathfinding problem (CHPP), and the disassembly line balancing problem (DLBP). I also show that a solution for CSPP can be used directly for CHPP and DLBP as well.

- In Section 4.3, I present a new multistep method called the Q-table compression method, which integrates different state space reduction steps with a dynamic training sampling technique to deliver an adaptive policy iteration algorithm.
- In Section 4.4, I demonstrate the applicability of my algorithm to three selected use cases: a CSPP, a CHPP, and a DLBP example.
- In Section 4.5, I give a summary and conclusions, and I describe further research directions and open research questions.

Finally, Chapter 5 concludes the main results of my research work due to the following sections:

- In Section 5.1, I give a final overview of the results of my work and collect future research questions and ideas that contain high potential.
- In Section 5.2, I present the thesis findings in an explicit form.
- In Section 5.3, I point out the applicability and usability of my scientific results from a broader aspect.

I hope the reader will enjoy my dissertation and find it interesting and valuable.

Chapter 2

Reinforcement learning in Industry 4.0

2.1 Introduction to Industry 4.0 applications

This chapter will present an overview of reinforcement learning methods in Industry 4.0 applications. On the one hand, there are several overviews of reinforcement learning methods from a theoretical point of view next to the fundamental book on the topic [4]. On the other hand, a detailed semantic overview of the Industry 4.0 frameworks [5] and a categorization of the research fields of Industry 4.0 are also described. A summary of key elements of Industry 4.0 research and several application scenarios [6] highlighted the broad scope of smart manufacturing. Although there is a lack of an extensive review of the Industry 4.0 revolution in different aspects, several relevant articles are available on this topic [7]. A survey on the applications of optimal control to scheduling in production, supply chain and Industry 4.0 systems [8] focused primarily on principle-based studies. Most industry 4.0 surveys and review articles declare the importance of optimization, but only general approaches are usually discussed, and no detailed guidelines are extracted. A comprehensive survey in the field of Industry 4.0 and optimization [9] discussed recent developments in data fusion and machine learning for the industrial prognosis, emphasizing identifying research trends, niches of opportunity, and unexplored challenges. Even if it considered several ML methods and algorithms, RL was mentioned only briefly without extracting its key fundamentals.

All of this above strengthened the motivation to prepare a detailed overview of RL applications and methods in the field of Industry 4.0 [R1]. My main goals with this were:

- presenting a hands-on reference for researchers who are interested in RL applications;
- giving compact descriptions of applicable RL methods;
- serving as a guideline to support them in quickly identifying the best fitting subset of RL methods to their problems, letting them focus on the relevant part of the literature.

My systematic review is based on examining the literature available from Scopus following PRISMA-P (Preferred Reporting Items for Systematic reviews and Meta-Analysis Protocols). The PRISMA-P workflow contains a 17-item checklist that facilitates the preparation and reporting of a robust protocol in a standardized way for systematic reviews. The literature source list was queried in February 2021 with the following keywords: `TITLE-ABS-KEY ("reinforcement learning" AND ("smart factory" OR "IOT" OR "smart manufacturing" OR "industry 4.0" OR "CPS"))`.

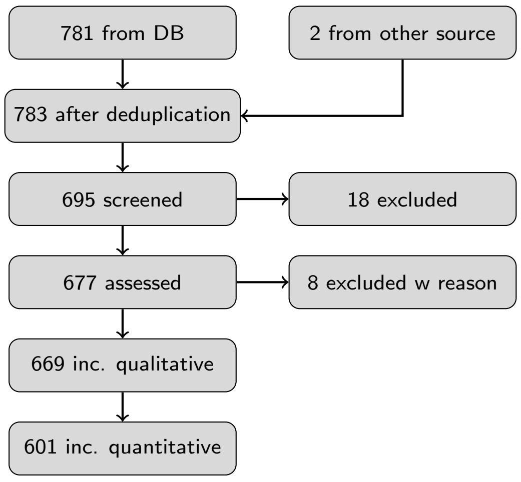


Figure 2.1: PRISMA processing flow

Both the author keywords and the index keywords were involved in the analysis. The keyword processing started with an extensive data cleansing process:

- by building up a standardized keyword unit (SKU) list and splitting complex keywords into SKUs

- by assigning SKUs to one of the following keyword classification types:
  - Principle captured
  - Industrial field of application
  - Application field of solution
  - Mathematical approach of application methodology
- by identifying major classification groups by classification types

781 articles were involved in the analysis. Of 14,035 original author and index keywords, 2,579 duplications were filtered out. The remaining 11,456 keywords were sliced into 45,824 SKUs. Finally, 12,017 keywords were assigned to classification types that provide the main tendencies and relations of industrial applications of reinforcement learning methods. Figure 2.1 shows the change in the size of the assessed literature in the PRISMA steps.

2.2 Theoretical background of Reinforcement Learning

In this section, I will summarize the fundamental concept of reinforcement learning and present a general classification of RL methods.

There are three main paradigms in machine learning: supervised, unsupervised, and reinforcement learning [9]. In supervised learning, the learner machine is trained on a labeled dataset, which stands for multi-dimensional input data records x_i and the assigned scalar outcomes y_i. The goal is to find the best fitting model f that describes the relationship between the input vectors and the outcome scalars: $y_i = f(x_i)$. In unsupervised learning, the training dataset is still required, but it contains unlabeled input records without output scalars. The learner machine aims to discover assignment rules or hidden patterns between different subsets of input dimensions or identify clusters in the input data where the elements within each cluster are very similar to each other but significantly different from the elements of other clusters. In reinforcement learning, the agent learns directly from the consequence of its decision, no preliminary training dataset is given. Therefore, the agent uses a hit-and-trial method to discover the environment and achieve the best available outcome [10].

In summary, supervised and unsupervised learning requires a preliminary dataset with or without labeled records. In contrast, reinforcement learning can only get results iteratively. Another significant difference is that in supervised learning the learner machine tries to find the overall best fitting model, while in reinforcement learning the agent tries to find the optimal point to reach the best outcome.

Reinforcement learning (RL) solves problems due to sequential learning. As Figure 2.2 represents the process, an agent takes observations of the environment and, based on that, executes an action (A_t). As a result of the action in the environment, the agent will get a reward (R_t) and can take a new observation (O_t) from the environment, and the cycle is repeated. The problem is to let the agent learn to maximize the total reward. The reinforcement learning concept was introduced in [4, Section 3.1]. While in supervised and unsupervised learning, the model fitting requires a complete set of observations, in reinforcement learning, the learning process is sequential.

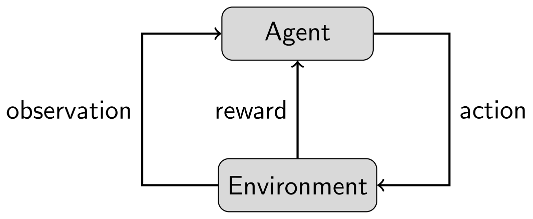


FIGURE 2.2: Reinforcement Learning framework

Reinforcement learning is based on the reward hypothesis, which states that all goals can be expressed as the maximization of expected cumulative rewards. Formally, the history is defined as the sequence of observations, actions, and rewards: $H_t = O_1, R_1, A_1, \dots, A_{t-1}, O_t, R_t$.

The state is the information used to determine what happens next. Formally, state is a function of history: $S_t = f(H_t)$. Let G_t denote the total discounted reward from the time step t: $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

The state-value function $v(s)$ gives the expected total discounted return if starting from state s: $v(s) = \mathbb{E}[G_t|S_t = s]$. The policy drives the agent's behavior in all possible cases, so it is essentially a map of states to actions. There are two main categories in it: 1) the deterministic policy: $a = \pi(s)$, 2) the stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$. The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π: $q_\pi(s; a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$.

Practically, the state-value function is a prediction of the expected present values (PV) of future rewards that allows to evaluate the goodness of states, so it is a map from states to scalars: $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$. The optimal state-value function $v^*(s)$ is the maximum state-value function in all policies: $v^*(s) = \max_\pi v_\pi(s)$. It is easy to find that if an optimal state-value function is known, then an optimal action-value function and an optimal policy can be derived.

The concept of reinforcement learning is based on stochastic processes and Markov chains. The Markov property is fundamental to the mathematical foundation of reinforcement learning methods. A state is Markov if and only if the $\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$ condition holds. By definition, a Markov Decision Process (MDP) is a tuple of $\langle \mathcal{S}; \mathcal{A}; \mathcal{P}; \mathcal{R}; \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$, $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$, γ is a discount factor, $\gamma \in [0; 1]$ and t time-steps are discrete. The Bellman equation practically states that the state-value function of an MDP can be decomposed into two parts: immediate reward and the discounted value of the successor states: $v(s = S_t) = R_{t+1} + \gamma v(S_{t+1})$.

Environments can be distinguished by their observability. Let denote S^a_t the agent's state at time-step t, S^e_t the environment's state. The environment can be 1) fully observable if the agent directly observes all states of the environment ($O_t = S^a_t = S^e_t$), or partially observable if the agent has indirect observations ($S^a_t = (\mathbb{P}[S^e_t = s_1], \dots, \mathbb{P}[S^e_t = s_n])$).

2.2.1 Reinforcement learning methods

In this section, I describe the major methods of reinforcement learning one by one, highlighting their properties and evolutionary stages, following David Silver's approach from the most simple to the most complex.

2.2.1.1 Multi-armed bandit problem

The multi-armed bandit problem (MABP) is one of the simplest introductory problems in reinforcement learning. An agent repeatedly chooses from k options or actions. After each choice, it receives a numerical reward based on a stationary distribution corresponding to its selection. The objective is to maximize the total rewards collected in a given time period or in a limited number of steps. The problem was named in analogy to a slot machine, or 'one-armed bandit', except that it has k levers instead of one.

Obviously, if the q^* action-value function is known, the k-armed bandit problem would be trivially solvable. Therefore, those scenarios should focus on cases where action values are not known with complete certainty, although they may have estimates. The core challenge lies in adequately balancing exploration and exploitation. During exploration, the agent gathers information about the rewards associated with previously unselected actions, whereas, during exploitation, it selects the action that appears optimal based on its current knowledge.

If exploration is chosen too frequently, the agent may accumulate irrecoverable losses or miss out on potential gains. Conversely, if exploration is chosen too infrequently, the agent may fail to discover actions with higher rewards and, as a result, suffer long-term losses. Various sophisticated methods exist to balance exploration and exploitation, ensuring that, under certain conditions, the agent's performance converges to the maximum attainable total reward. However, these conditions are not always guaranteed to hold in practical applications. Nevertheless, the k-armed bandit problem provides a prominent and fundamental framework for understanding the trade-off between exploration and exploitation.

2.2.1.2 Dynamic programming

Dynamic programming (DP) covers a decision process by breaking it down into a sequence of elementary decision steps over time. "Dynamic" refers to the sequential approach, while "programming" refers to its optimization objective.

In this Section, all the methods work assuming the environment is perfectly known. The iterative policy evaluation method is described for learning the state-value function of a given policy π. The value iteration method is used to determine the optimal state-value function. However, actions are taken according to any given policy π, and last but not least, the policy iteration is presented to derive an optimal policy to the environment.

Dynamic programming (DP) algorithms are generally limited because they assume that they know the environment perfectly and have high computational requirements. However, dynamic programming methods provide the essence of the ideas used in advanced techniques in an easily understandable form.

Iterative policy evaluation

Assume that a policy π is given and actions are taken according to it. The goal is to determine the state-value function v_π by iterative application of the Bellman backup: $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_\pi$. At each and every iteration step, the state-value function should be updated in the following way:

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right) \tag{2.1}$$

On the basis of the single Bellman decomposition, the $\pi(a|s)$ term provides the probability of taking action a by following policy Π. The $\mathcal{R}_s^a$ term shows the discounted direct reward from state s by taking action a, while $v_k(s')$ term shows the upcoming cumulative rewards, and finally $\mathcal{P}_{ss'}^a$ describes the probability of the trajectory occurring. It can be proven that with weak conditions, the proposed state-value function update will converge to $v_\pi(S)$ [4, Section 4.2].

Value function iteration

The iterative policy evaluation method can be extended to find an optimal state-value function $v^*(s)$. The main idea behind that iteration should be done by starting from the final reward and working backwards. Assume that the solution of subproblem $v^*(s')$ is known. Then by the solution of next iteration step $v^*(s)$ can be found by one-step look-ahead:

$$v^*(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \right) \tag{2.2}$$

It can be easily seen that for finite state space $\mathcal{S}$, the determination of optimal state-value function for all the available states can be done in a finite number of steps [4, Section 4.4].

Policy iteration

The fundamental concept of policy iteration is to determine the optimal policy by alternating between policy evaluation and policy improvement. The goal is to find the best possible action-selection strategy that maximizes the expected cumulative reward. In the policy evaluation step, a fixed policy π is given. The agent computes the corresponding state-value function $v_\pi(s)$ by solving the Bellman equation iteratively, as discussed earlier. This step determines how good a given policy is by estimating the expected rewards when following it. Then, in the step of policy improvement, the agent uses the given current value function $v_\pi(s)$ to update the policy by choosing actions that maximize the expected reward:

$$\pi'(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \right) \tag{2.3}$$

If the policy stabilizes, so no further changes occur, the algorithm has converged to an optimal policy π^* [4, Section 4.3]. It has been proven that policy iteration is guaranteed to converge to an optimal policy in finite Markov Decision Processes (MDPs) [11]. The policy iteration learning process is demonstrated in Figure 2.3.

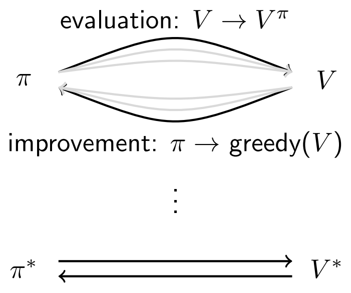


FIGURE 2.3: Learning by policy iteration method

Asynchronous dynamic programming

Asynchronous dynamic programming (ADP) is a variation of traditional dynamic programming methods that update different parts of the state-value function at other times rather than updating all states simultaneously. This approach allows for more flexible and scalable computations, making it suitable for large-scale problems where full sweeps of the state space are computationally infeasible. ADP methods update states in an arbitrary order, often prioritizing updates where the most significant changes occur. These methods still rely on the Bellman equation but can converge faster by focusing on the most needed computational effort.

2.2.1.3 Model-free prediction methods

Unlike dynamic programming, model-free methods do not require a perfectly known environment. Instead, they need only experience samples or sequences of states, actions, and rewards, with no prior knowledge of the environment.

This section presents the Monte-Carlo learning method based on averaging the experience. Then, the Temporal-Difference learning method is discussed to let the agent learn by more frequent but smaller steps by applying bootstrapping techniques. In contrast, the Temporal-Difference(λ) learning method extends the Temporal-Difference method's one-step learning to multiple-step learning.

Monte-Carlo learning

The Monte-Carlo (MC) agent solves the RL problem by applying the average sample return to learn from complete episodes. Hence, episodes must be guaranteed to be terminated because otherwise, the learning process cannot be performed. MC uses the most straightforward idea by assigning the empirical mean of the returns to a specific state [4, Section 5.1]. There are two main types of MC methods:

- *First-visit MC*: only a state's first visit will be involved in the calculation during an episode. Assume that state s is visited for the first time in time period t. Denote G_t the total return from the time period t and $N(s)$ the number of times that state s is visited, while $S(s)$ the sum of G_t returns up to the current episode. Then the estimate of the state-value will be the empirical mean: $V(s) = S(s)/N(s)$. As the experience grows, as $N(s) \to \infty$, the long-term mean will converge to the state-value function: $V(s) \to v_\pi(s)$.
- *Every-visit MC*: all visits of a state will be involved in the calculation during an episode. Formally, the main difference from the first-visit MC is that $N(s)$ needs to be incremented in each time period t whenever the state s is visited.

From a computational point of view, it is important to mention that the empirical mean is determined incrementally in practice. Denote $V^{(n)}(s)$ the estimate of the value-function, while $S^{(n)}(s)$ the cumulative sum of returns after episode n then $G_t^{(n)}$ the total return in episode n from time period t when state s is visited and assume that state s is visited kth times overall.

$$V^{(n)}(s) = \frac{1}{k}S^{(n)}(s) = \frac{1}{k}\sum_{i=1}^{n} S^{(i)}(s) = \frac{1}{k}\sum_{i=1}^{n-1} S^{(i)}(s) + \frac{1}{k}G_t^{(n)} = \tag{2.4}$$

$$= \frac{1}{k}(k-1)V^{(n-1)}(s) + \frac{1}{k}G_t^{(n)} = V^{(n-1)}(s) + \frac{1}{k}\Big(G_t^{(n)} - V^{(n-1)}(s)\Big)$$

Figure 2.4 shows the learning process of the Monte-Carlo method. It can be easily seen that the learning step is performed at the end of an episode.

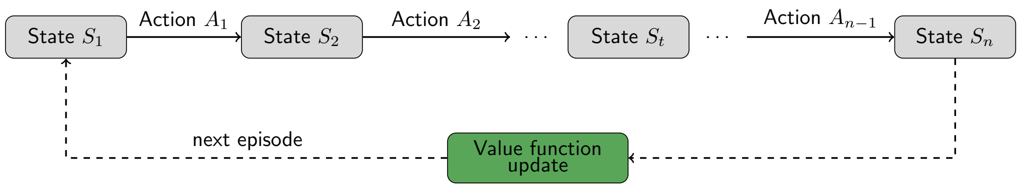


FIGURE 2.4: Monte-Carlo learning method

Temporal-Difference learning

The Temporal-Difference (TD) agent learns from incomplete episodes by applying bootstrapping. Compared to MC learning, TD uses the best guess of the total return, or formally $R_{t+1} + \gamma V(S_{t+1})$ instead of episodic experience G_t to calculate the values' estimates $V(s)$. This single difference indicates that the TD agent can perform a learning step after each and every action [4, Section 6.1], as Figure 2.5 shows. As a consequence, it can be applied to never-ending episodes.

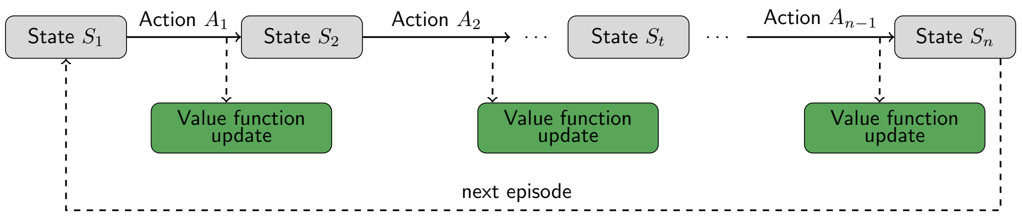


FIGURE 2.5: Temporal-Difference learning method

Temporal-Difference(λ) learning

There are intermediate solutions between TD that perform updates of the value function (VF) estimate after a one-step return and MC that performs updates only at the end of an episode (practically ∞-step return). The main idea behind this approach is to apply the normalized geometric series $(1-\lambda)\lambda^{n-1}$ to weight the n-step returns $G_t^{(n)}$ [4, Section 7.1]. In this case, the estimate of the value function will use a weighted total return of $G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$. It can be shown that TD(0) is equivalent to every-visit MC learning, and TD(1) is equivalent to the original TD learning methods. Furthermore, TD(λ) methods can be applied both forward and backward. The algorithms shown in this section can be used whether

- in offline mode: value function estimate updates are accumulated within episodes but applied only at the end of the episode or
- in online mode: value function estimate updates are accumulated within episodes and can be applied immediately.

A unified view of model-free prediction techniques is shown in Figure 2.6. First, it was created by Richard Sutton, but David Silver prepared this version. It highlights the two most important dimensions of learning methods: the vertical dimension represents the depth of the updates, while the horizontal dimension represents the width of the updates.

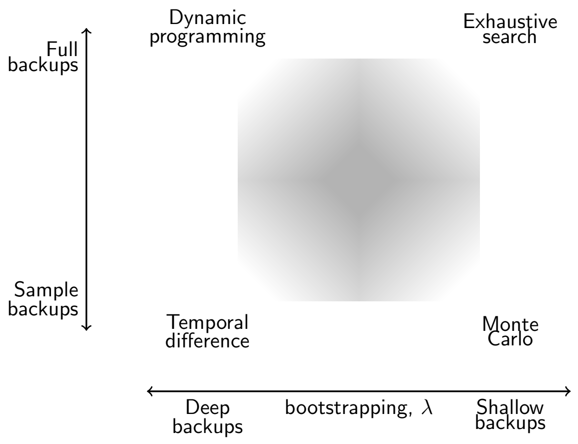


FIGURE 2.6: Unified view of model-free prediction techniques

2.2.1.4 Model-free control methods

In the previous Section, model-free prediction methods were summarized. These methods learn from others' experiences, so acting policies are managed externally and called off-policy learning. In contrast, on-policy learning allows the algorithm to take actions based on its own policy. Hence, a primary objective steps forward: to optimize the policy.

In this section, the ϵ-Greedy policy iteration is described to combine the exploitation of current knowledge of optimal decisions and the exploration of unknown new potentials. Furthermore, the on-policy temporal-difference control method, known as the SARSA method, is presented by applying bootstrapping techniques to speed up the learning process.

ϵ-Greedy policy iteration control

ϵ-Greedy policy iteration covers a combined solution. On the one hand, the MC method is applied to learn the action-value function $Q(s;a)$. On the other hand, the agent can act greedily, meaning it will choose the optimal action based on the actual action-value function $Q(s;a)$. This kind of action policy exploits only the current experience and does not support the exploration of alternatives. With a slight change in the strategy, this kind of issue can be solved: let the agent act randomly with probability ϵ, and greedily with probability $(1-\epsilon)$ [4, Section 5.4]:

$$\pi(a|s) = \begin{cases} 1-\epsilon+\frac{\epsilon}{m} & \text{if } a = \underset{a'\in\mathcal{A}}{\operatorname{argmax}}\ Q(s,a') \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases} \tag{2.5}$$

On-policy temporal-difference control method, aka SARSA method

Similarly to model-free prediction methods, an algorithm lets the agent learn from incomplete episodes by applying bootstrapping [4, Section 6.4]. In this case, the ϵ-Greedy policy iteration method needs to be modified in the following way: instead of using the MC method, TD learning should be applied to learn the action-value function $Q(s;a)$ that allows one to perform a learning step after each and every action and acting according to the most updated action-value function in a way similar to that of the ϵ-Greedy policy iteration. The SARSA name comes from an acronym: State s $\rightarrow$ Action a $\rightarrow$ Reward r $\rightarrow$ State s' $\rightarrow$ Action a'. Following the SARSA method, the update of the action-value function should look like: $Q(s;a) \leftarrow Q(s;a) + \alpha\Big(r + \gamma Q(s';a') - Q(s;a)\Big)$. It can be proved that, under certain conditions, SARSA action-value function converges to the optimal action-value function: $Q(s;a) \rightarrow q^*(s;a)$.

2.2.1.5 Off-policy learning

There are several situations in which the learning process is not based only on its own experience. Formally, this means that the target policy $\pi(a|s)$ or state-value function $v_\pi(s)$ or action-value function $q_\pi(s;a)$ is determined by observing the results of an external behaviour policy $\mu(a|s)$.

This section shows the importance sampling (IS) to determine the most accurate learning objective. Then, Q-learning is described as an effective alternative for obtaining a function iteration with a lower variance.

Importance sampling

One possible way to handle the difference between the target and behaviour policies is importance sampling when a correction multiplier is applied by processing observations [4, Section 5.8]. If MC learning is combined with importance sampling, then the update of the value-function will look like: $(S_t) \leftarrow V(S_t) + \alpha\Big(G_t^{\pi/\mu} - V(S_t)\Big)$. But because corrections are made at the end of an episode, the product of multipliers can drive a dramatically high variance. Hence, MC learning is not suitable for off-policy learning.

Therefore, TD learning seems much more adequate to combine with importance sampling, because the correction multiplier should be applied for only a single step and not for a whole episode:

$$(S_t) \leftarrow V(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\Big(R_{t+1} + \gamma V(S_{t+1})\Big) - V(S_t)\right) \tag{2.6}$$

Q-learning

Another possible way to handle the difference between the target and behaviour policies is to modify the value-function update logic as Q-learning does [4, Section 6.5]. Assume that in state S_t the very next action is derived by using behaviour policy: $A_{t+1} \sim \mu(\cdot|S_t)$. By taking action A_{t+1} immediate reward R_{t+1} and the next state S_{t+1} will be determined. But for value-function update, consider an alternative successor action based on target policy: $A' \sim \pi(\cdot|S_t)$. Therefore, importance sampling will not be necessary, and the Q-learning value function update will look like: $Q(S_t;A_t) \leftarrow Q(S_t;A_t) + \alpha\Big(R_{t+1} + \gamma Q(S_{t+1};A') - Q(S_t;A_t)\Big)$.

In a special case, if the target policy π is chosen as a pure greedy policy and the behaviour policy μ follows the ϵ-greedy policy, then the so-called SARSAMAX update can be defined as follows: $Q(S;A) \leftarrow Q(S;A) + \alpha\Big(R + \gamma \max_{a'} Q(S';a') - Q(S;A)\Big)$. Last but not least, it was proven that Q-learning control converges to the optimal action-value function: $Q(s;a) \rightarrow q_*(s;a)$.

2.2.1.6 Value function approximation

The Reinforcement Learning methods discussed in the previous sections represented value functions by lookup tables, but in practice, operating with state- or state-action-level lookup tables is not feasible. On the one hand, it would be very memory and computation-intensive, and on the other hand, the learning process would be too slow if the state and/or action spaces are large. The solution for such large problems is to estimate the state-value and action-value functions with function approximation: $\hat{v}(s;\mathbf{w}) \approx v_\pi(s)$ and similarly: $\hat{q}(s;a;\mathbf{w}) \approx q_\pi(s;a)$.

There are many kinds of function approximation methods that can be applied: linear combination of features, neural network, decision tree, and Fourier bases. In this Section, the first two types of methods are discussed. First, the gradient descent method is presented, which can be effectively combined with Monte-Carlo or Temporal-Difference methods for value function approximations. Then, the deep Q-network is described, serving as a more sample-effective learning method.

Value function approximation by stochastic gradient descent

A well-known tool for the function approximation is gradient descent [4, Section 9.3]. Denote $J(\mathbf{w})$ as a differentiable function of the parameter vector $\mathbf{w}$. Define the gradient of $J(\mathbf{w})$ as $\nabla_\mathbf{w} J(\mathbf{w}) = \left(\frac{\partial J(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial J(\mathbf{w})}{\partial w_n}\right)^T$. To find a local minimum of the parameter $J(\mathbf{w})$, $\mathbf{w}$ must be adjusted in the direction of the negative gradient by $\Delta\mathbf{w} = -\frac{1}{2}\alpha\nabla_\mathbf{w} J(\mathbf{w})$ where α is the learning step-size parameter.
A practical solution is to use gradient descent with a linear combination of features because the formulas become much simpler in this case. The representation of the value-function will look like: $\hat{v}(S;\mathbf{w}) = \mathbf{x}(S)^T\mathbf{w} = \sum_{i=i}^{n} x_i(S)w_i$, while the objective function to minimize the mean-squared error between the true value function and its approximation can be calculated using the formula of $J(\mathbf{w}) = \mathbb{E}_\pi\left[\Big(v_\pi(S) - \mathbf{x}(S)^T\mathbf{w}\Big)^2\right]$. It is proven that stochastic gradient descent with a linear combination of features converges to the global optimum. Furthermore, the update

rule is quite simple: $\nabla_{\mathbf{w}}\hat{v}(S;\mathbf{w}) = \mathbf{x}(S)$, and then $\Delta\mathbf{w} = \alpha\Big(v_\pi(S) - \hat{v}(S;\mathbf{w})\Big)\mathbf{x}(S)$. The result shows that the adjustment parameter $\mathbf{w}$ stands for three components: learning step-size $\cdot$ prediction error $\cdot$ feature value. In practice, the true value function is usually not known, but a noisy sample of it is known at different methods:

- for MC method, the target is G_t and hence parameter update:
  $\Delta\mathbf{w} = \alpha\big(G_t - \hat{v}(S_t;\mathbf{w})\big)\nabla_{\mathbf{w}}\hat{v}(S_t;\mathbf{w})$
- for TD(0) method, the target is the TD target $R_{t+1} + \gamma\hat{v}(S_{t+1};\mathbf{w})$ while parameter update:
  $\Delta\mathbf{w} = \alpha\big(R_{t+1} + \gamma\hat{v}(S_{t+1};\mathbf{w}) - \hat{v}(S_t;\mathbf{w})\big)\nabla_{\mathbf{w}}\hat{v}(S_t;\mathbf{w})$
- for TD(λ) the target is λ-return G_t^λ and parameter update:
  $\Delta\mathbf{w} = \alpha\big(G_t^\lambda - \hat{v}(S_t;\mathbf{w})\big)\nabla_{\mathbf{w}}\hat{v}(S_t;\mathbf{w})$

Whichever method is chosen, the RL learning process needs to update the value function approximation with the same frequency as the original method.

Deep Q-network

Even though gradient descent-based value function approximation methods can be very calculation-effective and updates can be managed incrementally, they are less sample-effective, which means that the information that could be extracted from an observation will not necessarily be exploited.

There are batch methods that work with experience replay (ER). Preliminarily, all observed experiences should be collected. Denote $\mathcal{D}$ the consisting experience of state-value pairs: $\mathcal{D} = \Big\langle \langle s_1; v_1^\pi\rangle, \ldots, \langle s_n; v_n^\pi\rangle \Big\rangle$. Artificial observations can be generated by random sampling from the history of experience: $\langle s; v^\pi\rangle \sim \mathcal{D}$ Therefore, stochastic gradient descent can be applied to it: $\Delta\mathbf{w} = \alpha\Big(v^\pi - \hat{v}(s;\mathbf{w})\Big)\nabla_{\mathbf{w}}\hat{v}(s;\mathbf{w})$. In this way, $\mathbf{w}^\pi$ converges to an optimal least-squares solution.

One of the most commonly used RL methods was born by combining experience replay and Q-learning with a periodically frozen target policy:

1. Using behavior policy, action a_t can be taken according to ϵ-greedy policy.
2. Transitions should be stored in replay memory $\mathcal{D}$ as $\langle s_t, a_t, t_{t+1}, s_{t+1}\rangle$.

3. Random minibatch samples of transitions (s, a, r, s') can be generated from $\mathcal{D}$
4. Based on them, Q-learning targets will be determined by using fixed parameters w^-
5. Minimize mean squared error between Q-network and Q-learning targets:

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a', w_i^-) - Q(s, a, w_i)\right)^2\right] \tag{2.7}$$

2.2.1.7 Policy gradient

In contrast to value-based methods where optimal action can be determined based on the learned value function in a particular state, policy gradient (PG) methods directly approximate the optimal policy:$\pi_\theta(s, a) = \mathbb{P}[a|s, \theta]$.

It is necessary to have an objective function $J(\theta)$ to measure the goodness of fitting policy π_θ to the optimal policy. In this case, policy-based RL becomes an optimization problem to find the optimal θ according to $J(\theta)$. Some methods use gradients such as gradient descent, conjugate gradient, or Quasi-Newton method, and others do not, such as hill climbing, simplex, or genetic algorithms. In general, these kinds of methods show better convergence properties and can work effectively with high-dimensional or continuous action spaces, and last but not least, they can learn stochastic policies. However, policy gradient methods typically converge to a local optimum rather than a global one. It is essential to note that the value functions can also be used to learn the optimal θ parameter. However, once learned, the value functions are unnecessary when selecting the optimal action.

Softmax

Let $J(\theta)$ be a policy objective function. Policy gradient descent algorithms search for a local optimum in $J(\theta)$ by ascending the gradient of the policy: $\Delta\theta = \alpha\nabla_\theta J(\theta)$ By assuming that policy π_θ is differentiable and its gradient is $\nabla_\theta \pi_\theta(s, a)$. Likelihood ratios can be transformed to the following form:$\nabla_\theta \pi_\theta(s, a) = \pi_\theta(s, a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} = \pi_\theta(s, a)\nabla_\theta \log \pi_\theta(s, a)$, where $\nabla_\theta \log \pi_\theta(s, a)$ is called score function.

The Softmax policy method is based on the approach of weighting actions using linear combinations of features $\phi(s, a)^T\theta$ [4, Section 13.2]. Therefore, the probabilities of actions are proportional to exponentiated weights: $\pi_\theta(s, a) \propto e^{\phi(s,a)^T\theta}$ The score function looks like the following: $\nabla_\theta \log \pi_\theta(s, a) = \phi(s, a) - \mathbb{E}_{\pi_\theta}[\phi(s, \cdot)]$.

Gaussian/Natural policy gradient

In continuous action spaces, Gaussian policy is a natural option. In this case, the mean is a linear combination of features: $\mu(s) = \phi(s,a)^T\theta$. Fixing the variance as σ^2, the policy will be Gaussian: $a \sim \mathcal{N}\Big(\mu(s), \sigma^2\Big)$. The score function will look like: $\nabla_\theta \log \pi_\theta(s,a) = \frac{1}{\sigma^2}\Big(a - \mu(s)\Big)\phi(s)$.

Monte-Carlo policy gradient method aka REINFORCE

Monte-Carlo policy gradient method, or, by a more popular name, the REINFORCE algorithm, updates the θ parameter by using stochastic gradient ascent. It is firmly based on the policy gradient theorem that generalizes the likelihood ratio approach to multi-step MDPs by replacing immediate reward r with long-term values of $Q^\pi(s,a)$ with weak restrictions on $J(\theta)$. The key idea behind that the locally optimal policy can be found by ascending the gradient of the objective function as follows: $\theta_{t+1} \leftarrow \theta_t + \alpha\nabla_{\theta_t} \log \pi_{\theta_t}(s_t,a_t)v_t$, where v_t is an unbiased sample of $Q^\pi_{\theta_t}(s_t,a_t)$.

Actor-critic policy gradient

In practice, REINFORCE still has a high variance. To handle it, the action value function can also be estimated: $Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$. In this way, there are two sets of parameters:

- Critic: updates action-value function parameters w
- Actor: updates policy parameters θ according the actual version of critic

Essentially, the actor tries to find the probability of all the available actions and select the best one. Meanwhile, the critic evaluates the selected action by estimating the value of the new state that results from performing the action.

2.2.1.8 Model-based methods

Model-free methods learn the value-function and/or policy directly from their experience of a real environment. Extending the experience collection process can increase the accuracy of RL knowledge. This can be achieved either by setting up an artificial virtual environment and defining reward and state transition functions that describe the real environment well or by building a model that approximates the real environment by learning its history.

If it is assumed that the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are known, then the model $\mathcal{M} = \langle \mathcal{P}_\eta; \mathcal{R}_\eta \rangle$ is a representation of MDP $\langle \mathcal{S}; \mathcal{A}; \mathcal{P}; \mathcal{R} \rangle$ if $S_{t+1} \sim \mathcal{P}_\eta(S_{t+1}|S_t, A_t)$ and $R_{t+1} = \mathcal{R}_\eta(R_{t+1}|S_t, A_t)$. The learning model from experience is a supervised learning problem. Figure 2.7 presents the basic concept of model-based learning methods.

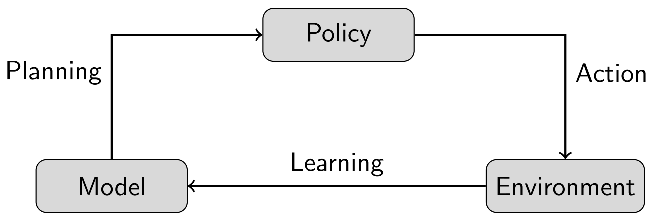


FIGURE 2.7: Model-based reinforcement learning process

First, the model should learn, and therefore, an internal simulation environment can be defined. Then, the model representation and the model-free RL methods can be used. So, model-based techniques differ from model-free ones in using internal model representation to derive rewards and state transitions.

Prioritized sweeping

Prioritized sweeping (PS) improves value iteration by prioritizing updates to states that are expected to have the most significant impact on future decisions. Instead of performing uniform updates across all states, it maintains a queue of priority updates and processes them in order of significance.

The priority of a state s is typically based on the temporal difference error: $p(s) = |r + \gamma \max_{a'} Q(s', a') - Q(s, a)|$

Prioritizing sweeping significantly improves the convergence speed compared to standard value iteration by focusing on the most important updates and skipping unnecessary ones [4, Section 8.4].

DYNA

DYNA is a hybrid RL approach that combines model-free and model-based learning. It uses real experiences to learn both a model of the environment and an

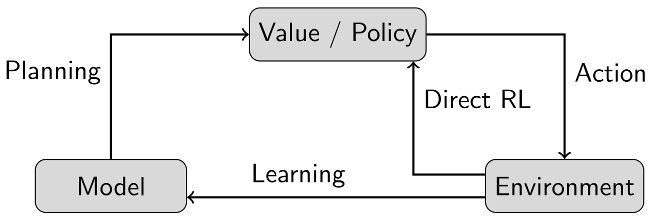


FIGURE 2.8: DYNA framework

action-value function. The model is then used to generate simulated experiences, which are used to update the value function, accelerating learning.

As Figure 2.8 shows, the DYNA framework involves:

- Direct RL updates: Learning from real experiences.
- Learning: Estimating state-transition probabilities and rewards.
- Planning with Simulated Experiences: Using the learned model to generate additional training data.

By integrating these components, DYNA enables faster learning compared to purely model-free methods.

Imitation Imitation learning (IL) is an RL paradigm where an agent learns to make decisions by mimicking expert demonstrations rather than learning purely through trial and error. This approach is particularly useful when designing a reward function is difficult or when exploration-based learning is too costly or unsafe.

There are two main types of imitation learning:

1. *Behavioural cloning (BC)*: This is a supervised learning approach in which an agent directly maps states to actions based on expert demonstrations. The agent learns a policy $\pi(s)$ from given expert trajectories (s_t, a_t) that approximates the expert's decision-making process: $\pi(a|s) \approx \pi^*(a|s)$. Behavioural cloning is simple and efficient but suffers from compounding errors: mistakes can propagate if the agent encounters states not covered in the given expert data.

2. *Inverse Reinforcement Learning (IRL)*: Instead of directly learning a policy, IRL infers a reward function from expert demonstrations and then optimizes a policy based on this learned reward function. The main advantage of IRL is its ability to generalize beyond the specific actions demonstrated by the expert. Still, it is computationally expensive and often requires solving multiple reinforcement learning problems.

Figure 2.9 summarizes a classification of the discussed reinforcement learning methods in a tree structure.

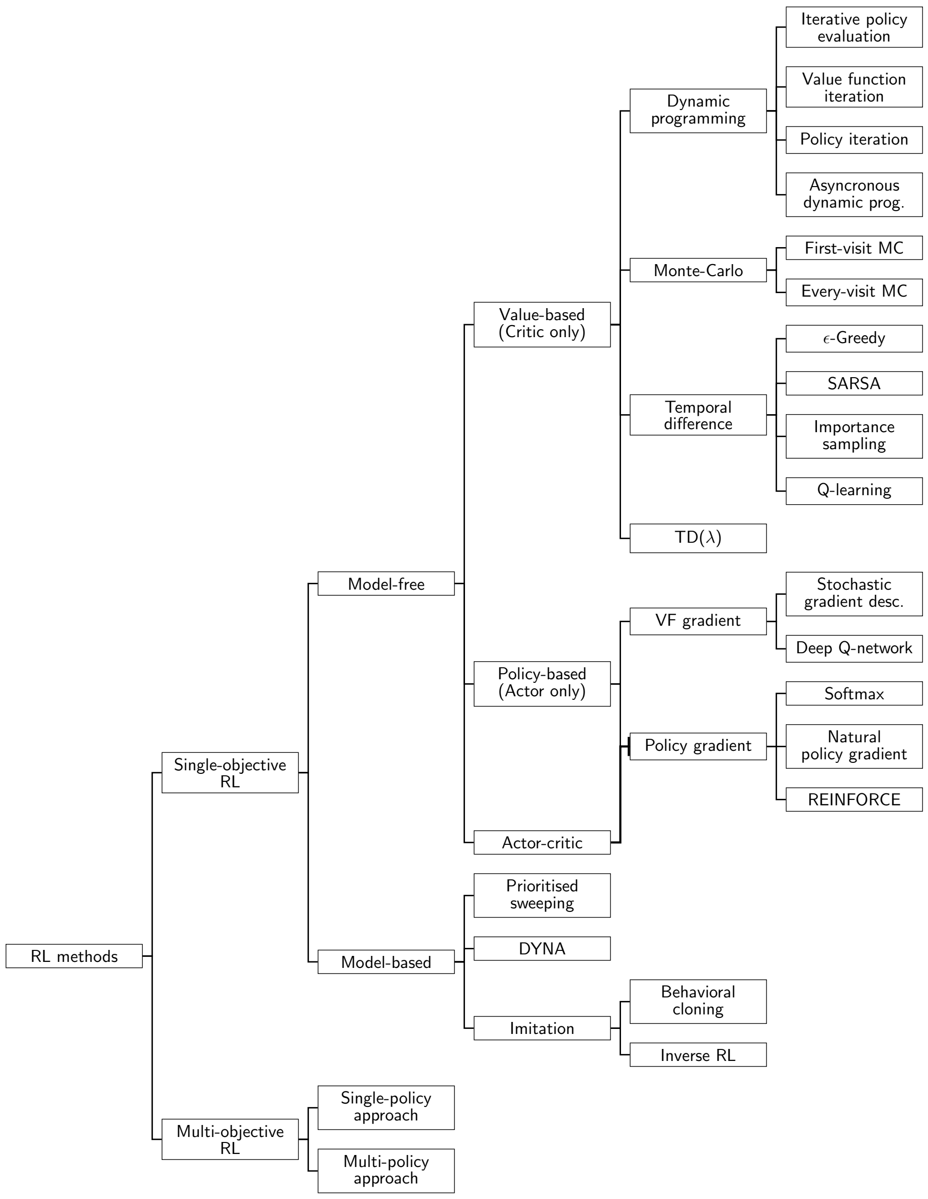


FIGURE 2.9: Classification tree of reinforcement learning methods

2.2.1.9 Multi-agent learning systems

At Industry 4.0 applications, no single RL agent is usually set up, but multiple ones. Multi-agent reinforcement learning (MARL) topic addresses the sequential decision-making problem of numerous autonomous agents operating in a common or quite similar environment, each of which aims to optimize its own long-term return by interacting with the environment and a central system and/or other agents.

Markov games

One way to generalize MDPs for applying multiple agents is Markov games (MG), also known as stochastic games. Formally, the Markov game can be defined as a tuple $\left\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{R^i\}_{i\in\mathcal{N}}, \gamma \right\rangle$, where $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of agents $N > 1$, $\mathcal{S}$ denotes the state space of all agents, and $\mathcal{A}^i$ denotes the action space of agent $i \in \mathcal{N}$. Introducing $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$, let $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ the transition probability function from any state $s \in \mathcal{S}$ to a particular state $s' \in \mathcal{S}$ for a joint action of $a \in \mathcal{A}$, while $R^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function that determines the immediate reward by starting from state s, taking action a and moving to state s'. Lastly, $\gamma \in [0, 1)$ is the discount factor. Figure 2.10 shows the general framework of Markov games.

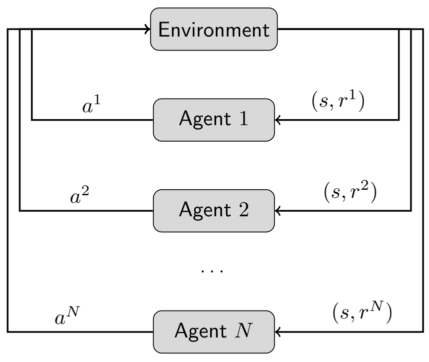
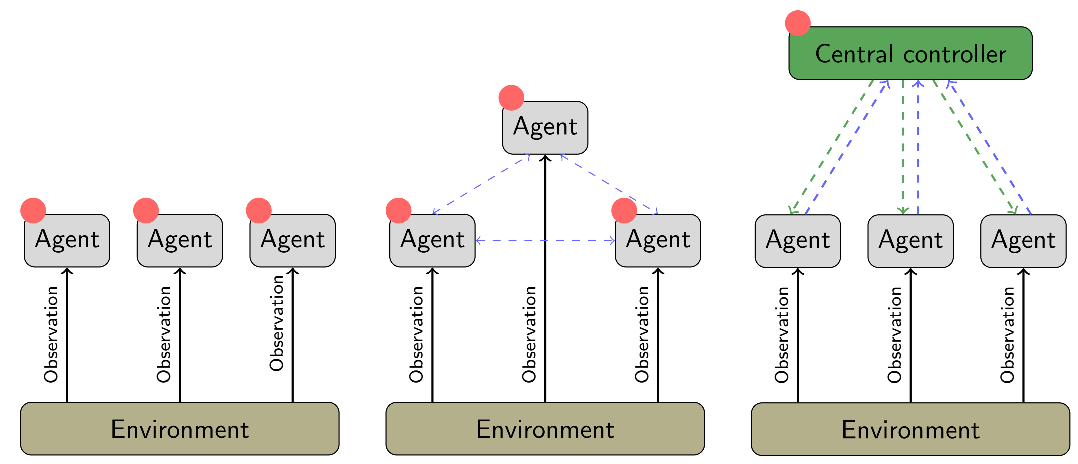


FIGURE 2.10: Schematic diagram of Markov games

Markov games extend traditional reinforcement learning to environments where multiple agents interact, either competitively or cooperatively. Each agent learns its own policy while considering the presence and actions of other agents. There are multiple basic ways to organize learning:

- local learning, no centralized knowledge (see Figure 2.11a)
- local knowledge deployment, local learning, central knowledge collection
- local knowledge deployment, local learning with knowledge transfer to close neighbourhoods (see Figure 2.11b)
- local knowledge deployment, centralized learning (see Figure 2.11c)



(A) Full local learning (B) Knowledge sharing (C) Centralized learning

● Policy improvement - - › Observation sharing - - › Policy sharing

FIGURE 2.11: Cooperation concepts of multi-agent learning systems

The complexity arises due to non-stationarity—since the environment dynamics change as other agents learn. To resolve this problem it can be assumed that all agents have the same or identical reward function in a fully cooperative setting: $R^1 = R^2 = \cdots = R^N = R$. This is also referred to as multi-agent MDP (MMDP). With this approach, the state- and action-value functions are identical to all agents, which thus enables the single-agent RL algorithms to be applied if all agents are coordinated as one decision maker. The global optimum for cooperation now constitutes the Nash equilibrium of the game.

Nash equilibrium (NE) characterizes an equilibrium point π^*, from which none of the agents has any incentive to deviate. As a standard learning goal for MARL, NE always exists for discounted MGs but may not be unique in general. Most of the MARL algorithms are contrived to converge to such an equilibrium point.

There are multiple approaches for using MARL setup:

Distributed MARL

In distributed multi-agent reinforcement learning, agents learn in a decentralized fashion, each updating their own policies independently, often in parallel. This setup is particularly useful for large-scale problems where centralized coordination is impractical due to communication constraints or computational costs.

Each agent follows its own policy $\pi_i(a|s)$ and updates it based on local experiences without direct synchronization with other agents. Distributed learning methods commonly use experience replay across agents or asynchronous policy updates to improve sample efficiency and scalability [4, Section 15.10].

Cooperative MARL

Cooperative MARL focuses on optimizing a shared team objective rather than individual rewards. Agents must work together to achieve a common goal, often modelled as a partially observable Markov game (POMG). A popular method is Centralized Training with Decentralized Execution (CTDE), where agents are trained together but act independently using learned policies.

A key challenge in cooperative learning is the credit assignment problem, where it is difficult to determine which agent's actions contributed to a positive or negative outcome. Techniques such as difference rewards or multi-agent actor-critic (MAAC) help improve coordination.

Collaborative MARL

Collaborative MARL is a broader concept where agents not only cooperate but also actively share knowledge to accelerate learning. Unlike traditional cooperative methods where agents optimize a shared objective, collaborative learning allows agents to exchange experiences or gradients to improve sample efficiency.

One approach is experience sharing (ES), where agents periodically share trajectories to help others learn from different perspectives. Another method is policy distillation, where multiple agents transfer their learned policies into a single shared model.

Centralized MARL

In centralized MARL, a single learning process optimizes the policies of all agents together, often using a shared value function or a centralized critic. The goal is to optimize a global objective rather than individual rewards. Centralized learning can be formulated as:

$$Q(s, a_1, ..., a_N) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right] \tag{2.8}$$

where $Q(s, a_1, ..., a_N)$ represents the joint action-value function for all agents.

This approach benefits from full observability and coordination but suffers from scalability issues as the number of agents increases.

Federated MARL

Federated reinforcement learning is a privacy-preserving approach where multiple agents learn collaboratively without directly sharing raw data. Instead, they train local policies and periodically communicate model updates to a central server that aggregates the policies into a global model:

$$\pi^{\text{global}} = \frac{1}{N}\sum_{i=1}^{N} \pi_i^{\text{local}} \tag{2.9}$$

This is particularly useful in scenarios where data privacy is critical. A challenge in federated RL is ensuring stability in policy aggregation, as different agents may have vastly different local environments and objectives.

This summary of the major reinforcement learning methods gave a valuable and efficient overview of the concept behind it. As my literature overview shows, there are numerous further modifications and extensions in addition to the basic methods. Later in this chapter, I will present a questionnaire in Figure 2.15 to ensure the appropriate methodology to set up an RL solution and to determine appropriately the relevant classes of RL methods that can provide a suitable solution to be fitted to their learning problems.

2.3 Overview of the Industry 4.0 relevant applications

In this section, I will present hands-on references in tabular format based on my data cleansing process results and some significant outcomes of systematic literature analysis. These will highlight some general trends that can help the reader successfully apply RL methods by preventing inappropriate trials and thereby shortening development periods. In the final part of the section, I will present a hands-on guideline to summarize the key conclusions.

2.3.1 Classification of applications by principle captured

The main goal of this section is to give an overview of the principal captured problem types for which reinforcement learning was applied, to describe the major tools that gave an impressive performance for each and every problem category, and finally, to highlight some typical issues that need to be taken care of during the implementation.

I identified the most relevant keywords assigned to a captured principle by performing an SKU analysis. Table 2.1 lists the associated publications by principal categories captured.

Furthermore, Figure 2.12 shows the principal classes captured by the reinforcement learning methods. The frequency table behind in Table 2.2 does not fulfil the essential criteria of the χ^2-test. Still, it lets identify some significant deviation from the overall distribution of RL methods by the captured principle classes.

- In the class of Prediction, Forecasting, Estimation, and Planning, the value function approximation methods and Markov decision processes are overrepresented. This concludes that the less complex methods are the focus, fully in line with a better understanding of the environment's behaviour without strong optimization aims.
- In the class of Detection, Recognition, Prevention, Avoidance, Protection, the policy gradient methods are overrepresented, while MDPs are underrepresented. This shows that researchers are more interested in complex models with higher predictive performance than basic solutions.

TABLE 2.1: Publication reference by principle captured

Principle captured	Referred publications
Prediction, Forecasting, Estimation, Planning	[12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41]
Detection, Recognition, Prevention, Avoidance, Protection	[42], [43], [12], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [31], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [41], [73]
Evaluation, Assessment	[74], [75], [20], [76], [77], [78], [56], [79], [80], [81], [68], [34], [82], [83]
Classification, Clustering	[84], [44], [85], [86], [87], [88], [89], [90], [37], [71], [91], [68], [92], [93], [83], [94], [71], [95], [96], [97]
Decision making	[98], [99], [84], [13], [100], [15], [101], [45], [102], [103], [19], [104], [105], [22], [23], [106], [107], [108], [109], [26], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [68], [91], [124], [71], [64], [125], [126], [63], [127], [128], [95], [129], [130], [131], [132], [133], [40]
Allocation, Assignment, Resource management	[98], [134], [99], [135], [136], [137], [138], [139], [140], [141], [142], [143], [144], [145], [101], [85], [146], [102], [147], [148], [149], [150], [151], [105], [152], [22], [153], [106], [47], [154], [106], [155], [156], [157], [158], [24], [156], [158], [159], [160], [161], [162], [163], [164], [165], [115], [166], [167], [168], [169], [77], [170], [121], [171], [33], [172], [173], [89], [174], [175], [176], [132], [177], [178], [179], [123], [180], [181], [182], [127], [183], [184], [185], [186], [187], [188], [133], [189], [129], [190], [69], [191], [192], [193], [194], [195], [80], [196], [62], [197], [132], [198], [199], [67], [93], [72], [200], [34], [41], [123], [180], [181], [201], [202], [203], [198], [93], [204], [133]
Scheduling, Queuing, Planning	[98], [205], [206], [74], [207], [14], [90], [208], [101], [21], [209], [152], [153], [23], [210], [211], [162], [212], [112], [26], [115], [213], [214], [89], [215], [216], [93], [217], [133], [127], [186], [218], [219], [34], [220], [221], [95], [124], [222], [223], [224], [190]
Control	[225], [42], [226], [227], [228], [207], [14], [229], [101], [16], [17], [103], [230], [151], [231], [20], [152], [232], [107], [233], [234], [235], [108], [236], [109], [25], [237], [29], [238], [114], [239], [240], [113], [241], [242], [243], [244], [245], [246], [247], [248], [249], [169], [250], [251], [171], [33], [170], [252], [253], [173], [254], [247], [255], [191], [256], [257], [258], [219], [259], [260], [201], [132], [72], [261], [262], [38], [95], [71], [39], [125], [202], [190], [58], [124], [263], [93], [182], [264], [131], [265], [266], [267]

- In the classes of Evaluation, Assessment, Allocation, Assignment, and Resource Management, the multi-agent methods are more in focus, which shows that this field is on the way to distributing the tasks to lower-level tools instead of centralized data processes. However, while in the first class, the distribution of further RL methods follows the overall distribution, in the second class, the policy gradient methods are overrepresented, which comes from the fact that allocation-related problems prefer to create an optimal policy.

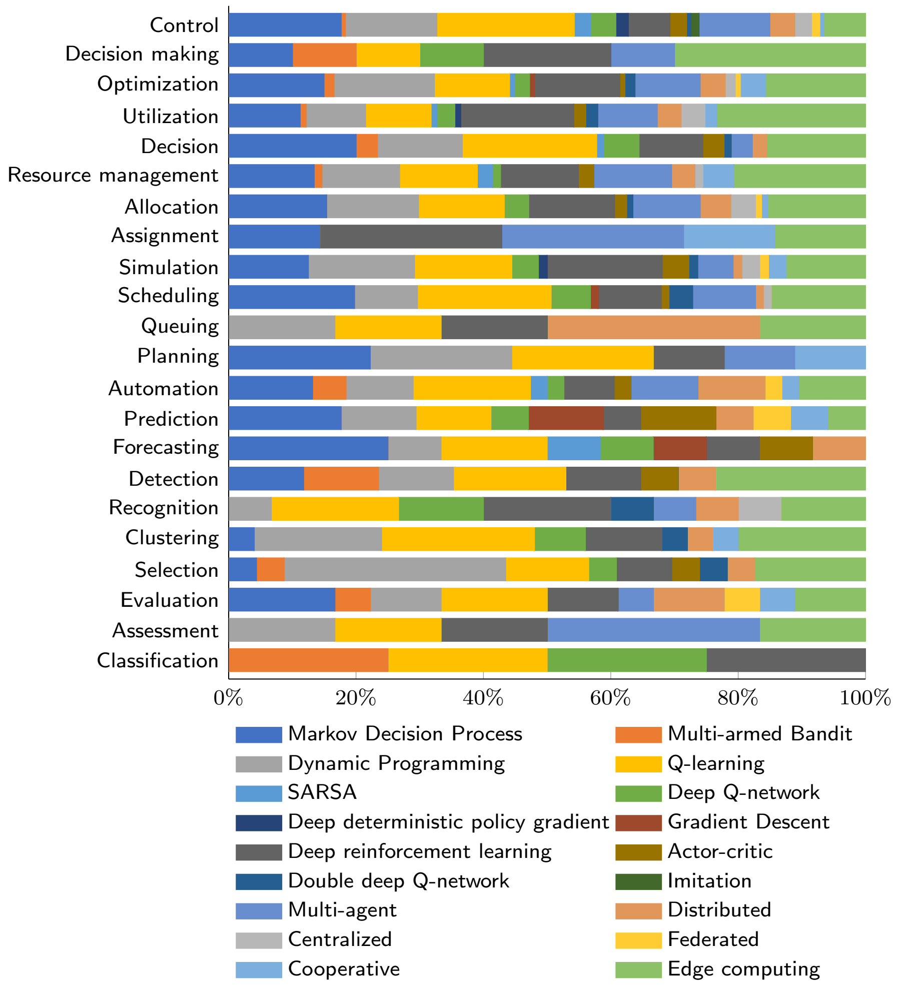


FIGURE 2.12: Distribution of RL methods by principal captured classes

- In the classes of Classification, Clustering, Decision-making, Scheduling, Queuing, and Planning, the situation is the opposite: multi-agent methods are underrepresented, which means that research of these kinds of operations is still focusing on a centralized solution.

- In the class of Control the Temporal Difference methods and Markov Decision Process contractions and multi-agent methods are overrepresented, while complex approaches, like policy gradient methods, are underrepresented.

- Discussions of specific parts of RL solution design problems occur in fewer cases. However, these types of publications demonstrate that building an appropriate RL application is not always trivial. I should highlight the state space design [226], [14], [229], [146], [268], [210], [109], [27], [222], [35], [269], [181], [219], [224], [195] and Action space design [111], [270], [248], [222],

TABLE 2.2: χ^2-test table of principle captured classes by RL method types

Principle captured	Markov decision process	Multi-armed Bandit	Dynamic	Temporal difference	Value function approximation	Policy gradient	Multi-agent	Edge computing
Prediction, Forecasting, Estimation, Planning	1.96	-0.58	0.12	-0.32	3.08	0.31	-0.39	-4.17
Detection, Recognition, Prevention, Avoidance, Protection	-3.09	1.51	-1.11	-0.17	0.39	2.2	-1.38	1.65
Evaluation, Assessment	-0.82	0.63	-0.08	-0.62	-1.21	-0.6	2.96	-0.27
Classification, Clustering	-3.61	0.56	1.28	1.41	1.54	0.65	-2.88	1.05
Decision making	3.1	2.47	-0.84	1.73	0.95	0.02	-10.82	3.39
Allocation, Assignment, Resource management	-2.69	-1.95	0.21	-11.18	-4.74	2.08	10.54	7.74
Scheduling, Queuing, Planning	2.17	-1.33	-2.17	1.24	1.61	-0.04	-2.63	1.16
Control	2.67	-1.34	2.35	7.52	-1.72	-4.93	4.27	-10.82

Reward construction [271], [228], [16], [272], [112], [273], [274], [248], [78], [275], [201], [222] and Exploration strategy planning [276], [88] which can be determinants from the whole application point of view.

2.3.2 Classification of publications by industrial field of application

Similarly, as shown in Section 2.3.1, I also identified the most relevant keywords assigned to industrial fields by performing SKU analysis. The associated publications are listed by industrial field categories in Table 2.3.

Similarly to the categories of principles presented earlier, I also prepared Figure 2.13 showing the industrial field classes by reinforcement learning methods. The frequency table behind in Table 2.4 does not meet the essential criteria of the χ^2-test. Still, it allows for identifying some significant deviation from the overall distribution of RL methods by industrial field classes.

TABLE 2.3: Publication reference by industrial field of application

Industrial field	Referred publications
Energy, Solar, Power, Electric	[277], [206], [135], [136], [227], [74], [138], [278], [228], [229], [16], [85], [279], [280], [281], [146], [102], [147], [103], [19], [282], [20], [232], [283], [268], [233], [47], [234], [284], [210], [285], [156], [286], [236], [287], [288], [109], [289], [110], [77], [163], [237], [290], [212], [161], [25], [291], [292], [293], [294], [242], [245], [295], [87], [167], [173], [248], [296], [215], [169], [55], [90], [170], [120], [33], [77], [250], [297], [298], [299], [181], [39], [261], [194], [72], [300], [301], [223], [302], [34], [177], [303], [53], [304], [83], [254], [179], [305], [306], [125], [307], [308], [309], [202], [310], [200], [224], [311], [259], [218], [185], [176], [312], [178], [190], [82], [257], [266], [267]
Telecommunication, Communication, Networking, Internet, 5G, Wi-Fi, Mobile	[225], [313], [277], [205], [135], [314], [84], [136], [315], [74], [138], [139], [316], [43], [14], [143], [317], [144], [145], [208], [85], [279], [280], [281], [147], [102], [148], [19], [318], [319], [320], [104], [209], [153], [321], [47], [106], [245], [322], [323], [108], [24], [286], [236], [324], [288], [159], [325], [162], [291], [111], [161], [77], [290], [289], [238], [326], [292], [327], [328], [114], [329], [330], [241], [331], [293], [332], [168], [30], [333], [247], [167], [334], [52], [335], [294], [336], [89], [169], [90], [213], [216], [171], [77], [122], [173], [296], [337], [170], [299], [172], [307], [190], [40], [123], [338], [176], [202], [180], [260], [52], [339], [177], [41], [340], [92], [181], [72], [35], [37], [341], [192], [97], [342], [203], [199], [198], [93], [67], [250], [189], [343], [344], [312], [311], [308], [256], [179], [302], [53], [345], [254], [183], [220]
Wireless, Radio, Antenna, Signal	[225], [313], [277], [98], [205], [314], [84], [137], [74], [138], [43], [142], [143], [317], [208], [346], [85], [16], [279], [281], [102], [147], [318], [319], [320], [104], [20], [153], [232], [209], [347], [106], [348], [349], [47], [323], [108], [48], [286], [236], [160], [288], [211], [325], [290], [110], [237], [162], [111], [332], [109], [331], [241], [238], [350], [330], [274], [166], [250], [245], [335], [334], [295], [333], [168], [30], [87], [246], [247], [172], [336], [351], [170], [352], [88], [175], [337], [253], [70], [259], [307], [343], [219], [309], [180], [72], [353], [338], [181], [93], [345], [190], [203], [250], [254], [302], [53], [311], [342], [306], [257], [301], [220], [256], [202], [34], [217], [218], [35], [66], [130], [133], [200], [96], [310], [193], [304], [40]
Vehicle, Unmanned aerial vehicle, Drone, Aircraft	[225], [313], [315], [12], [354], [279], [230], [231], [153], [268], [234], [355], [322], [284], [163], [327], [274], [246], [172], [93], [219], [356], [70], [259], [307], [312], [311], [81], [261], [183], [343], [92], [199], [185], [301], [338], [222]
Cyber Physical System, Robot	[226], [227], [45], [75], [17], [151], [231], [268], [23], [284], [235], [76], [211], [109], [357], [290], [111], [114], [358], [240], [359], [244], [243], [251], [249], [250], [270], [360], [352], [191], [124], [133], [68], [361], [58], [356], [260], [71], [52], [307], [190], [266]
Manufacturing, Factory	[207], [101], [281], [230], [151], [21], [349], [211], [362], [249], [38], [71], [363], [95], [361], [130], [364], [81], [125]
City, Building	[365], [206], [142], [366], [20], [47], [284], [357], [237], [294], [367], [246], [334], [215], [368], [33], [342], [203], [369], [68], [370], [176]

- In the class of Energy, Solar, Power, and Electric, the applications of Q-learning methods are over-represented, while more basic methods and policy gradient methods are under-represented.
- In the class of Telecommunication, Communication, Networking, Internet,

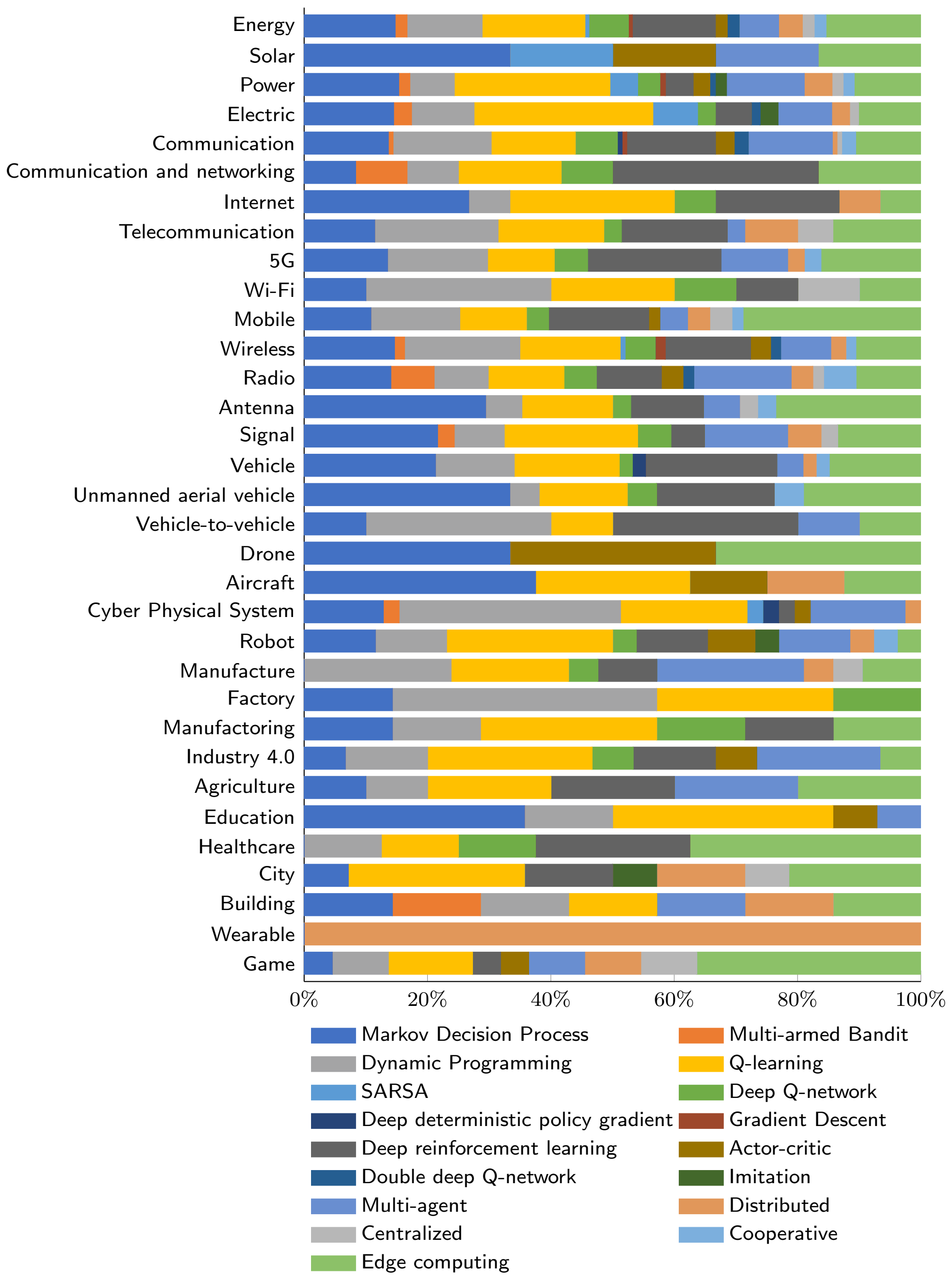


FIGURE 2.13: Distribution of RL methods by industrial field classes

5G, Wi-Fi, Mobile, the policy gradient methods are over-represented, and there is a strong focus on edge computing applications.

- In the class of Wireless, Radio, Antenna, and Signal, the applications of Markov Decision Processes are highlighted.

TABLE 2.4: χ^2-test table of industrial field classes by RL method types

Principle captured	Markov decision process	Multi-armed Bandit	Dynamic	Temporal difference	Value function approximation	Policy gradient	Multi-agent	Edge computing
Energy, Solar, Power, Electric	0.52	1.7	-12.77	21.87	0.94	-11.54	2.05	-2.77
Communication, Networking, Internet, 5G, Wi-Fi, Mobile	-8.61	-3.51	6.29	-18.79	2.23	13.24	-3.14	12.29
Wireless, Radio, Antenna, Signal	5.77	3.07	-1.73	-6.62	1.33	-1.76	2.68	-2.73
Vehicle, Unmanned aerial vehicle, Drone, Aircraft	8.44	-1.39	-2.32	-2.89	-2.49	5.9	-6.94	1.68
Cyber Physical System, Robot	-1.75	0	8.14	3.86	-2.23	-2.14	1.97	-7.86
Manufacturing, Factory	-3.33	-0.55	4.16	1.36	1.23	-2.54	1.52	-1.84
City, Building	-1.05	0.69	-1.77	1.21	-1.01	-1.17	1.87	1.23

- Similarly, in the class of Vehicle, Unmanned aerial vehicle, Drone, and Aircraft, the applications of Markov Decision Process are over-represented together with policy gradient methods, while the multi-agent solutions are less discussed.
- In the classes of Cyber-Physical Systems, Robot and Manufacturing, and Factory, the basic Dynamic methods and Q-learning approaches are more popular.
- Finally, in the class of City, Building the multi-agent methods are over-represented.

2.3.3 Classification of publications by mathematical approach of application methodology

Similarly, as shown in the previous sections, I also performed the SKU analysis for the third major dimension of keywords, which is the methodological approach to the solution. The most relevant keywords were identified, and then in Table 2.5, the associated publications are listed by methodological approach categories.

Although it is not feasible to summarize all the different methodological approaches in detail, I would like to highlight some specialities of selected cases to demonstrate how widely RL approaches are used and motivate researchers to find a solution for their problems from a new perspective.

As I described in Section 2.2, reinforcement learning methods are based on the Markov property, and hence, it is fundamental to model problems as Markov decision processes (MDPs), which is so far not trivial in several cases. By formulating an MDP man needs to take care of state space design, especially guaranteeing that a state representation contains all the relevant information to evaluate a situation, or with other words, anytime when the system is in the same particular action, the environment will take its response by the same characteristic for a specific action [98], [205], [315], [106], [348], [193].

Actor-critic methods are model-free learning methods that learn both the optimal policy for taking action and the value function for the most accurate evaluation of the current state. Most of the publications discuss mainly distributed autonomous IoT device networks. In these cases, the focus is shifted towards the learning and knowledge transfer solutions:

- Stochastic cloud-based IoT model for fog computing computation offload and radio resource allocation. [99]
- Centralized joint resource allocation solution for handling shortage of frequency resources of cellular systems by using a neural network embedded reinforcement learning algorithm. [178]
- Determine optimal sampling time for IoT devices for energy harvesting by saving batteries. Hence, state space contains continuous quantities, a linear function approximation was used, and a set of novel features was introduced to represent the large state space. [351]
- A bio-inspired RL modular architecture, called the transfer expert RL (TERL) model, can perform skill-to-skill knowledge transfer. Its architecture is based on an RL actor-critic model where both actor and critic have a hierarchical structure inspired by the mixture-of-experts model. [394]

Table 2.5: Publication reference by methodological approaches

Approach	Referred publications
Markov decision process	[98], [205], [135], [315], [140], [14], [146], [102], [103], [283], [209], [106], [233], [348], [155], [322], [86], [26], [25], [371], [372], [293], [276], [373], [167], [274], [216], [169], [214], [213], [172], [77], [254], [259], [374], [311], [375], [376], [129], [377], [193], [72], [201], [179], [378], [66], [222], [132], [258], [260], [342], [39], [219], [345], [265], [190], [261], [266]
Multi-armed Bandit	[379], [104], [380], [353], [68], [200], [63]
Dynamic programming	[134], [137], [315], [143], [280], [18], [147], [282], [21], [209], [349], [323], [157], [158], [211], [109], [86], [291], [163], [290], [238], [29], [164], [373], [359], [333], [244], [122], [54], [121], [214], [249], [256], [193], [308], [72], [92], [70], [342], [200], [203], [374], [381], [95], [306], [382], [224], [191], [260], [261]
Q-learning	[225], [205], [226], [135], [227], [207], [12], [229], [208], [85], [147], [103], [46], [19], [282], [349], [383], [154], [233], [284], [323], [210], [285], [26], [112], [212], [49], [328],[358], [329], [293], [246], [244], [250], [355], [384], [252], [214], [118], [297], [385], [368], [174], [298], [386], [248], [296], [369], [374], [70], [203], [93], [264], [94], [66], [126], [189], [376], [345], [129], [131], [224], [217], [96], [82], [221], [181], [185], [83], [52], [387], [388], [72], [182], [256], [127], [266]
SARSA	[16], [282], [242], [386], [248], [129]
Deep Q-network	[135], [85], [101], [19], [349], [49], [212], [293], [296], [389], [256], [127], [192]
Deep deterministic policy gradient	[231], [262], [340]
Gradient descent	[390], [28], [30], [218]
Deep reinforcement learning	[225], [313], [365], [277], [136], [43], [90], [100], [354], [102], [149], [319], [105], [21], [22], [209], [283], [348], [156], [357], [86], [325], [212], [49], [161], [77], [238], [167], [243], [391], [52], [335], [296], [119], [170], [392], [393], [262], [304], [201], [181], [382], [377], [222], [223], [195], [97], [79], [275], [184], [307], [263], [363], [301], [133], [62], [92], [303], [311], [178], [203], [343], [340], [34]
Actor-critic	[99], [17], [394], [19], [239], [351], [122], [178], [198], [340], [35], [339], [223]
Double deep Q-network	[85], [49], [212], [296], [389], [256], [127], [395]
Imitation	[228], [357], [267]
Multi-agent	[227], [396], [346], [147], [379], [282], [105], [233], [211], [325], [291], [397], [332], [165], [247], [253], [175], [170], [363], [369], [269], [221], [340], [220], [79], [72], [202], [34], [265], [177], [190], [62], [178], [197]
Distributed	[135], [396], [75], [147], [47], [284], [350], [121], [189], [93], [62], [58], [263]
Centralized	[149], [245], [298], [62], [189]
Cooperative	[346], [18], [172], [340], [202], [83]
Collaborative	[139], [383], [47], [239], [327], [398], [176], [198], [250]

- Deep reinforcement learning-based cooperative edge caching approach. [340]
- Multiple IoT devices send data parallel but generally do not provide additional information to the existing knowledge. So, it is not necessary to permanently send the data. By using Actor-Critic method, it can be determined which data packages need to be sent to prevent redundant or irrelevant communication [223]
- Mobile edge computing and energy harvesting framework of centralized training with decentralized execution by adopting MD-Hybrid-AC method. [122]
- Asynchronous Advantage Actor-Critic method for Mobile Edge Computing because computation offloading cannot perform well in many situations, but the optimal algorithm can be chosen for the IoT side. [198]
- Optimization of the robustness of IoT network topology with a scale-free network model which performs well in random attacks. A deep deterministic learning policy (DDLP) is proposed to improve stability for large-scale IoT applications. [339]
- IoT devices lack storage capacity, therefore a joint cache content placement and delivery policy for the cache-enabled D2D networks was constructed. [19]
- A federated reinforcement learning architecture was presented where each agent working on its independent IoT device shares its learning experience (i.e. the gradient of loss function) with each other. [239]

Centralized and federated methods

As Internet-of-Things (IoT) services and applications are growing rapidly, most current optimization-based methods lack a self-adaptive ability in dynamic environments. Learning-based approaches are generally implemented with a centralized setup to address these challenges. However, network resources may be over-consumed during the training and data transmission process. A federated deep-reinforcement-learning-based cooperative edge caching (FADE) framework is presented to solve complex and dynamic control issues. FADE enables base stations (BS) to cooperatively learn a shared predictive model by considering the first-round training parameters of the BS as the initial input to the local training and then uploading near-optimal local parameters to the BSs to participate in the next round of global training [18].

Although the first investigations have focused on designing learning algorithms with provable convergence time, other issues, such as the incentive mechanism, were explored later: a deep reinforcement learning-based incentive mechanism has been designed to determine the optimal pricing strategy for the parameter server and the optimal training strategies for edge nodes [149].

Hierarchical methods

Hierarchical approaches are primarily used to solve communication channel or information processing capacity issues. The model structure usually follows the structure of the information path. In a two-layer approach, a local IoT device transfers information to a local hub, which then transmits the collected data to the central decision-maker. In this case, separated models can be set up for both layers to find the optimal scheduling order for communication.

A new crowd-sensing framework is introduced based on a hierarchical structure to organize different resources, and is solved using a deep reinforcement learning-based strategy to ensure service quality [90]. A hierarchical correlated Q-learning approach (HCEQ) is presented to solve dynamic optimization of generation command dispatch (GCD) for automatic generation control (AGC) [233]. An enhanced version of a bioinspired reinforcement learning modular architecture is presented to perform skill-to-skill knowledge transfer, called the transfer expert RL (TERL) model. TERL architecture is based on an actor-critic RL model where both the actor and the critic have a hierarchical structure inspired by the mixture-of-experts model, formed by a gating network that selects experts who specialize in learning the policies or value functions of different tasks [394].

Another new cloud computing model was suggested for context-aware Internet of Things services, which is hierarchically composed of two layers: a cloud control layer (CCL) and a user control layer (UCL). The CCL manages cloud resource allocation, service scheduling, service profile, and service adaptation policy from a system performance point of view. Meanwhile, the UCL manages the end-to-end service connection and the service context from a user performance point of view. The proposed model can support non-uniform service binding and its real-time adaptation using meta-objects by intelligent service context management using a machine learning framework based on supervised reinforcement learning [152].

A new cooperative resource allocation algorithm that pairs reinforcement learning networks with prediction neural networks to track mobile targets accurately is presented. Specifically, a hierarchical structure that performs collaborative computing is designed to alleviate the computing pressure of front-end devices supported by edge servers [399]. A slightly different approach is applied to a resilient control problem studied for cyber-physical systems (CPSs) under the Denial-of-Service (DoS) attack. The term resilience is interpreted as the ability to be robust to external physical-layer disturbance and to defend against cyber-layer DoS attacks. The overall resilient control system is described by a hierarchical game, where the cyber security issue is modelled as a zero-sum matrix game, and a zero-sum dynamic game describes the physical minimax control problem. In virtue of the reinforcement learning method, the defence/attack policy can be obtained in the cyber layer. Additionally, the control strategy of the physical layer can be obtained using the dynamical programming method [400]. More publications on hierarchical RL topics are related to balancing timeliness and criticality when gathering data from multiple sources [118], ubiquitous user connectivity, and collaborative computation offloading for smart cities [250].

Distributed and parallel methods

It can be stated with certainty that intelligent devices have the most significant potential for industrial applications. In this context, intelligence means some kind of ability to make autonomous decisions and being able to perform learning steps locally. Considerable efforts have been made to develop functional solutions to achieve this goal.

Computation offloading can solve the problem of the high computational requirement of resource-constrained mobile devices. The mobile cloud is the well-known existing offloading platform, which is usually a far-end network solution. However, this can cause other issues, such as higher latency or network delay, which negatively affect real-time Internet of Things (IoT) mobile applications. Therefore, a deep Q-learning-based autonomic management framework is proposed as a near-end network solution for computation offloading at the mobile edge [135].

Another way to extend single reinforcement learning applications is to handle multiple objectives. There are two leading solution practices for handling such types of problems. The most obvious idea is to construct a mixed reward function that returns a combined result according to the different objectives [261], [372],

[163]. Another possible way is to combine multi-objective ant colony optimization methods with RL techniques such as deep reinforcement learning or double Q-learning algorithms [85], [144].

2.3.4 General trends of RL applications

Before the Industry 4.0 revolution began, the general methodology was based on centralized data collection, data processing, and predictive model development solutions. By spreading Internet of Things (IoT) devices, more computational tasks can be delegated to them. This potential is exploited by reacting to another significant issue: the lack of communication capability. On the one hand, communication between IoT devices and central servers or nodes is relatively energy-intensive. On the other hand, there are significant limitations on communication channels or frequencies.

Distributing computational tasks to IoT devices requires a fundamental change: It is impossible to assign as much human effort to data processing and predictive model development supervision as in the past. This was the main reason for appreciating the RL methods, which provide a general self-learning framework that basically requires no manual or human interactions to maintain.

Early research focused on the applicability of reinforcement learning techniques with single agents. Then, more and more complex problems were solved, and multi-agent solutions started to be analyzed. In recent years, the focus of the researchers has shifted to multi-agent structures. The setup of the agents and their goals or reward functions shows very creative solutions. In a new wave of research, the agents are defined with different roles, often with attacker-defender objectives, and each of the agents is trained in an optimal strategy according to it. Then, the stability and robustness of the system can be analyzed, and the weakest items can be purposely improved.

As Figure 2.14 shows, the number of research based on Industry 4.0-related reinforcement learning dynamically increases, and there is no sign of a slowing expectation in it. The number of publications included in my literature analysis is highlighted in blue on the chart.

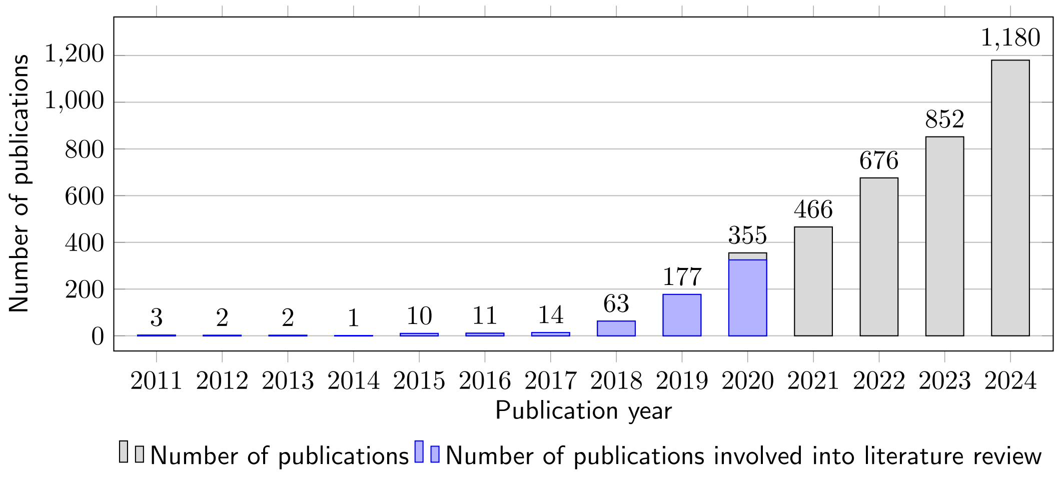


FIGURE 2.14: Industry 4.0-related RL publications by publication years

2.4 Discussion and guideline process to determine appropriate RL method to use

Based on the previous section, it can be highlighted that several reinforcement learning methods can be applied to Industry 4.0-related problems, and it is not trivial to determine which one provides a successful solution.

I prepared a questionnaire and presented it in a decision flow diagram in Figure 2.15 by using questions (Q), conclusions (C) and remarks (R). My primary goal was to establish a method to help readers formulate their RL tasks. The first questions of the questionnaire-based process verify whether the state and action spaces are appropriately defined and how the reward can be obtained. Further questions systematically narrow down the set of applicable RL methods. The possibility of using simulation or learning from one's own experience can determine the general learning mechanism. In contrast, the nature of reward propagation can determine a smaller subset of the RL methods that can be applicable. Using my guidelines, researchers will likely make fewer failed attempts, and the time to solution will be significantly reduced.

The reader should remember that the whole reinforcement learning concept is based on Markov Decision Processes. A direct conclusion is that state space should be constructed so that all potential states can contain all relevant information that can influence the outcomes. Moreover, the action space should be built similarly: the effects of an action in a particular state should be based on the same deterministic or stochastic behaviour. This will let the RL agent learn the effect mechanism behind it.

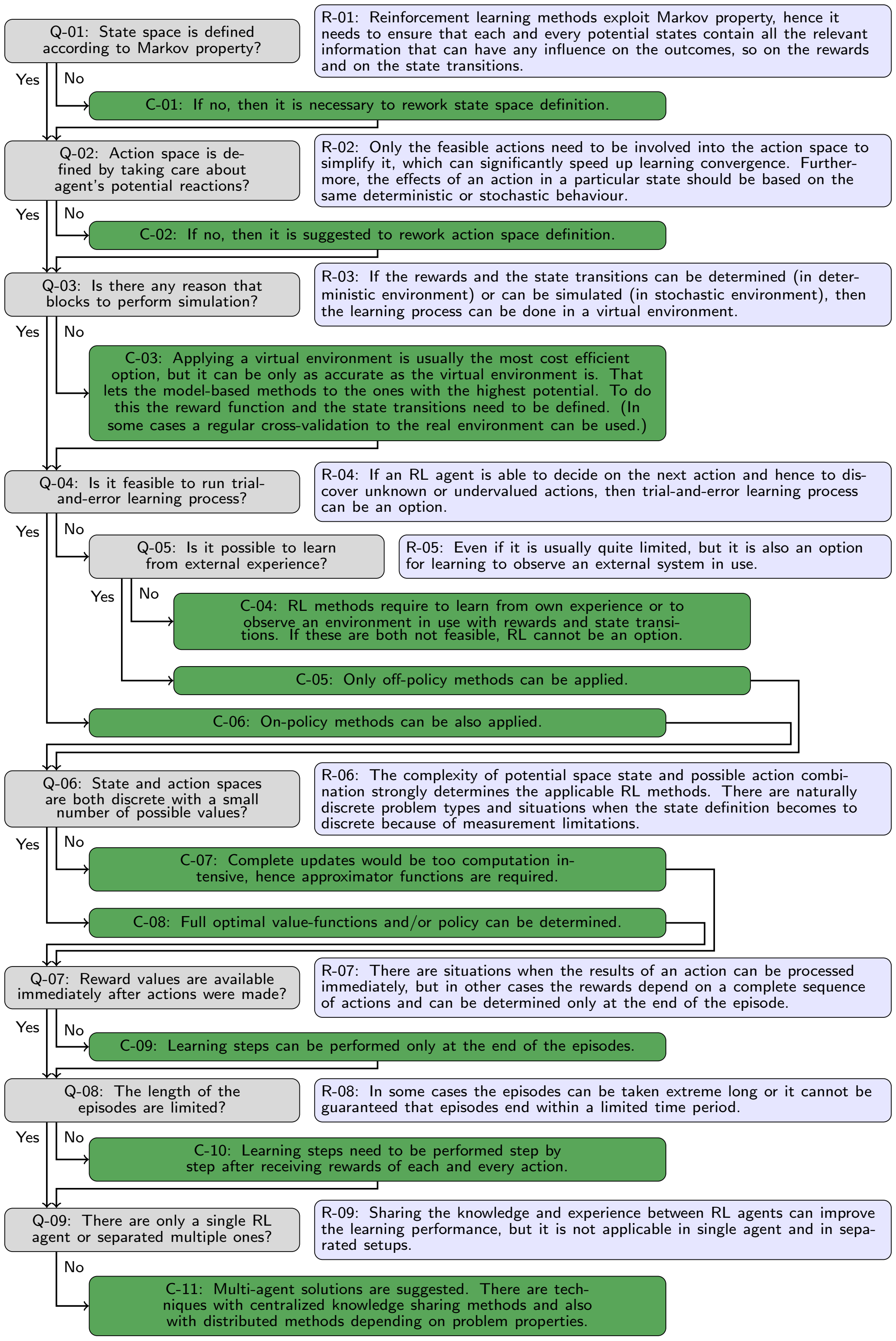

Q-01: State space is defined according to Markov property?
R-01: Reinforcement learning methods exploit Markov property, hence it needs to ensure that each and every potential states contain all the relevant information that can have any influence on the outcomes, so on the rewards and on the state transitions.
Yes
No
C-01: If no, then it is necessary to rework state space definition.
Q-02: Action space is defined by taking care about agent's potential reactions?
R-02: Only the feasible actions need to be involved into the action space to simplify it, which can significantly speed up learning convergence. Furthermore, the effects of an action in a particular state should be based on the same deterministic or stochastic behaviour.
Yes
No
C-02: If no, then it is suggested to rework action space definition.
Q-03: Is there any reason that blocks to perform simulation?
R-03: If the rewards and the state transitions can be determined (in deterministic environment) or can be simulated (in stochastic environment), then the learning process can be done in a virtual environment.
Yes
No
C-03: Applying a virtual environment is usually the most cost efficient option, but it can be only as accurate as the virtual environment is. That lets the model-based methods to the ones with the highest potential. To do this the reward function and the state transitions need to be defined. (In some cases a regular cross-validation to the real environment can be used.)
Q-04: Is it feasible to run trial-and-error learning process?
R-04: If an RL agent is able to decide on the next action and hence to discover unknown or undervalued actions, then trial-and-error learning process can be an option.
Yes
No
Q-05: Is it possible to learn from external experience?
R-05: Even if it is usually quite limited, but it is also an option for learning to observe an external system in use.
Yes
No
C-04: RL methods require to learn from own experience or to observe an environment in use with rewards and state transitions. If these are both not feasible, RL cannot be an option.
C-05: Only off-policy methods can be applied.
C-06: On-policy methods can be also applied.
Q-06: State and action spaces are both discrete with a small number of possible values?
R-06: The complexity of potential space state and possible action combination strongly determines the applicable RL methods. There are naturally discrete problem types and situations when the state definition becomes to discrete because of measurement limitations.
Yes
No
C-07: Complete updates would be too computation intensive, hence approximator functions are required.
C-08: Full optimal value-functions and/or policy can be determined.
Q-07: Reward values are available immediately after actions were made?
R-07: There are situations when the results of an action can be processed immediately, but in other cases the rewards depend on a complete sequence of actions and can be determined only at the end of the episode.
Yes
No
C-09: Learning steps can be performed only at the end of the episodes.
Q-08: The length of the episodes are limited?
R-08: In some cases the episodes can be taken extreme long or it cannot be guaranteed that episodes end within a limited time period.
Yes
No
C-10: Learning steps need to be performed step by step after receiving rewards of each and every action.
Q-09: There are only a single RL agent or separated multiple ones?
R-09: Sharing the knowledge and experience between RL agents can improve the learning performance, but it is not applicable in single agent and in separated setups.
No
C-11: Multi-agent solutions are suggested. There are techniques with centralized knowledge sharing methods and also with distributed methods depending on problem properties.

FIGURE 2.15: Guideline process to determine appropriate RL method to use

Once the state and action spaces are defined, it needs to be investigated whether performing simulations is an option or not. Suppose that the environment's behaviour can be determined when an action is made in a particular state by deriving the reward value and the state transition. In that case, an extensive learning process can be executed using model-based RL methods cost-effectively without a significant risk of applying untrained agents. The general rule is also valid: the RL solution will be as adequate as the simulation. If there is an option to validate the simulation results in the real environment, then this can help ensure the solution's adequacy.

Even if the questionnaire's conclusions are soft-defined, a user with some basic knowledge of RL methods can easily interpret them, or it can be a basis for an RL method selector wizard. An assignment of the conclusions to the RL methods is provided in Table 2.6 to simplify interpretation. It is important to emphasize that only methods previously presented are included in the assignment. Therefore, it is not exhaustive since many varieties of RL methods have already emerged.

TABLE 2.6: Assignments of conclusions to RL methods

Subject of conclusion	Suggested RL methods
Model-based methods	DYNA, Prioritized sweeping, Imitation
Off-policy methods	Dynamic programming, Monte-Carlo, Temporal difference (including *epsilon*-Greedy, SARSA, Importance sampling and Q-learning), TD(λ)
On-policy methods	Stochastic gradient descent, DQN, Softmax, Natural policy gradient, REINFORCE, Actor-critic
Discrete methods	Dynamic programming, Monte-Carlo, Temporal difference (including *epsilon*-Greedy, SARSA, Importance sampling and Q-learning), TD(λ), DQN
Updates only at end of episodes	Asyncronous DP, Monte-Carlo
Updates during episodes	Dynamic programming, Temporal difference (including *epsilon*-Greedy, SARSA, Importance sampling and Q-learning), TD(λ), Stochastic gradient descent, DQN, Softmax, Natural policy gradient, REINFORCE, Actor-critic

2.5 Summary of Industry 4.0 applications

In this chapter, I pointed out that reinforcement learning methods have a high potential in Industry 4.0 applications, which is a common agreement among researchers. One of the biggest reasons behind this is that smart tools require a high level of optimization that cannot be satisfied with human intervention. This continuously raises the demand for self-learning solutions, and RL techniques have proven their efficiency in multiple fields.

The primary goal of this chapter was to provide an overview of RL applications in the field of Industry 4.0. As a first step, I served a high-level overview of the general RL framework and a classification of RL methods to help facilitate the exploration of the possibilities. Then, a more detailed summary of the most widely used RL methods for Industry 4.0 applications was presented. Therefore, my result can serve as a starting point for further research on RL applications.

Then, I highlighted the results of my systematic literature overview of reinforcement learning applications in the field of Industry 4.0. An extensive keyword analysis led to identifying typical patterns by choosing an adequate RL method for particular combinations of principal captures and industrial fields. Although there are no unique optimal RL methods, there are RL methods that provide an efficient solution to some problems. My summary can be used as a hands-on reference for future research and help researchers shorten the preparation time for their research.

In addition, a questionnaire was prepared that provides a methodology for setting up the reinforcement learning system properly and choosing an appropriate method for the learning problem facing the researcher. I believe that an extension of my questionnaire can be the basis of a wizard tool that enables the user to find the most fitting RL method for the learning task and guide through the setup processes. On the other hand, by knowing the key properties of the different RL methods, adopting or modifying an existing one to fit the specific needs becomes faster, hence developing one's own RL method.

Chapter 3

Reinforcement learning in disassembly line balancing problem

3.1 Introduction to disassembly line balancing optimization

The increasing environmental pressures caused by human activities drive governments and organizations to limit or reduce their footprints. As an outcome, the European Commission declared this a strategic goal. A new Circular Economy Action Plan for a cleaner and more competitive Europe has been announced to sustainably transform the economy and society. There is a wide range of recommendations and proposals on how these types of issues should be handled, and the key point is to reduce consumption and increase the recycling rate. The efforts led to the creation of the concept of circular economy, which is more complex than just manufacturing optimization because it also covers the optimization steps of the supply chain, disassembly and recycling [401] [402]. A process flow overview diagram of the circular economy is shown in Figure 3.1 [403].

Every step of the circular process has practices and methods to optimize the operation. However, the primary goal of the circular economy is to optimize the whole supply chain to include recycling. The classical problem of optimizing the manufacturing supply chain is a thoroughly researched topic [404] [405] [406].

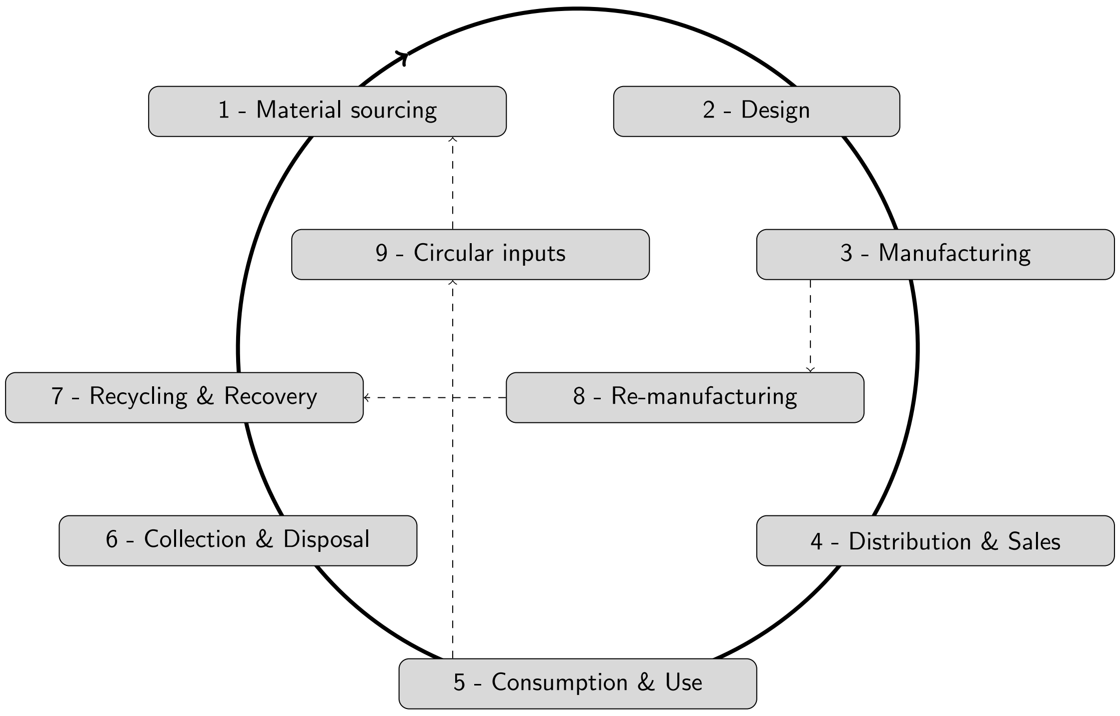


Figure 3.1: Schematic flow of circular economy [403]

The circular economy optimization goal also covers the recollection and recycling processes that can be a bit more complex because these are more stochastic and less controlled processes. The design of the disassembly line and its balancing problems describe the methodological background of the relevant segment of the whole circle [407] [408] [409]. The most basic models assumed deterministic inputs, but advanced models started handling the uncertainty observed in real problems. Among other things, stochastic variables describe the source materials and their distributions. Similarly, the disassembly tasks, their required process times, and the demand values of the recycling steps can also be stochastic attributes in some models.

The supply chain optimization and assembly line balancing topics have been intensively studied issues since the 1950's years, while the focused analysis of disassembly lines started almost 40 years later [410]. A detailed historical overview of disassembly methods [411] provided a summary table of the different solutions, which was extended using a comprehensive classification [412] in Table 3.1. Shows a wide range of machine learning methods for solving the disassembly line balancing problem. There are already successful RL-based solutions with attractive learning performance, but how to construct an effective implementation for a given problem needs to be explained in detail.

Table 3.1: Overview of disassembly line balancing solutions

Solution method	Author(s)	Year	Ref.	Speciality
Reverse MRP	S. Gupta, K. Taleb	1994	[413]	
Integer programming	D-H Lee, H-J Kim, G Choi, P Xirouchakis	2004	[414]	
Branch and bound	H-J Kim, D-H Lee, P. Xirouchakis, OK Kwon	2009	[415]	
Heuristic	K-N Taleb, S-M Gupta, L. Brennan	1997	[416]	Single product
Heuristic	K-N Taleb, S-M Gupta	1997	[417]	Multi-product
Petri-net	K-P Neuendorf, D-H Lee, D. Kiritsis, P. Xirouchakis	2001	[418]	
Ant colony optimization	Wang, Y. Rong, D. Xiang	2014	[419]	
Ant colony optimization	J. Wang, X. Wu, X. Fan	2015	[420]	Two-stage
Ant colony optimization	Y. Luo, Q. Peng, P. Gu	2016	[421]	Multi-layer
Ant colony optimization	H-E Tseng, C-C Chang, S-C Lee, Y-M Huang	2019	[422]	Hybrid bidirectional
Artificial bee colony	Y. Ren, G. Tian, F. Zhao, D. Yu, C. Zhang	2017	[423]	Multi-objective
Artificial bee colony	G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, J. Tan	2018	[424]	Dual-objective
Discrete bees algorithm	W. Xu, Q. Tang, J. Liu, Z. Liu, Z. Zhou, D-T Pham	2020	[425]	Human-robot collaboration
Genetic algorithm	T-F Go, D-A Wahab, M-A Rahman, R. Ramli	2010	[426]	
Genetic algorithm heuristic	J. L. Rickli, J. A. Camelio	2013	[427]	Multi-objective
Genetic algorithm	H-E Tseng, S-C Lee	2018	[428]	Interactive
Genetic algorithm	Y. Ren, C. Zhang, F. Zhao, H. Xiao, G. Tian	2018	[429]	Asynchronous
Genetic algorithm	Y. Tian, X. Zhang, Z. Liu, X. Jiang, J. Xue	2019	[430]	Cooperative
Genetic algorithm	J-C Chen, Y-Y Chen, T-L Chen, Y-C Yang	2022	[431]	Adaptive
Particle swarm opt.	W-D Li, K. Xia, L. Gao, K-M Chao	2013	[432]	Selective disassembly
Artificial fish swarm algo.	J. Guo, J. Zhong, Y. Li, B. Du, S. Guo	2018	[433]	Hybrid model
Teaching-learning-based opt.	K. Xia, L. Gao, L. Wang, W. Li	2013	[434]	Simplified method
Teaching-learning-based opt.	K. Xia, L. Gao, L. Wang, W. Li, X. Li, W. Ijomah	2016	[435]	Modified method
Graph-based method	S. Smith, G. Smith, W-H Chen	2012	[436]	Sequence structure graph
Graph-based method	P. Mitrouchev, C-G Wang, L-X Lu, G-Q Li	2015	[437]	Geometry contacting graph
Fuzzy model	X-F Zhang, G. Yu, Z-Y Hu, C-H Pei, G-Q Ma	2014	[438]	Fuzzy-rough set mapping
Fuzzy model	H-P Hsu	2016	[439]	Fuzzy knowledge-based
Reinforcement learning	E. Tuncel, A. Zeid, S. Kamarthi	2014	[440]	
Reinforcement learning	S. Mete, F. Serin	2021	[441]	

The above facts strengthened my motivation to prepare an effective RL-based self-learning solution that can be easily adapted to different disassembly problems [R2], [R3]. I will present a general guide on defining the reward function from the problem parameters. Furthermore, I will highlight the importance of customizing the action-taking method, which will be relevant whenever a general RL framework is parameterized. According to the current outlook, the number of disassembly lines and their optimization requirements will increase significantly, and such a self-learning solution will adequately explain this.

My first experiences with an RL-based solution show that the most basic version of training an RL agent is not obviously efficient: after declaring the state- and action spaces, the central task is to define the reward function. Most RL frameworks support this approach without the option of customizing the action-taking method. However, a large proportion of infeasible state-action pairs should not be learned. For disassembly line balancing problems, it is an obvious option to dynamically filter for the possible uncompleted actions by considering the precedence graph. Integrating such an efficient state-dependent action-restriction method can radically reduce the learning path. The same conclusion has been found [442]. The main idea was to apply identical constraints in formulating a mixed-integer quadratic problem (MIQP) and provide a systematic method to define an appropriate restriction for the action-selection step. The key result of this chapter is to present a Q-learning solution with an integrated heuristic for dynamic state-dependent action restriction. A disassembly line optimization procedure with the RL method named the dlOptRL algorithm is described and explained in detail. From a higher point of view, my algorithm belongs to the class of Heuristically Accelerated Reinforcement Learning (HARL) methods, which is proven by an appropriate formulation as well.

3.2 Optimization of disassembly line balancing

In this section, I will summarize the general notation of disassembly problems. Then, I will go through the solution methods, including the problem formulation as a linear programming task.

3.2.1 Problem formulation

Consider a disassembly line balancing model for a single product with finite supply. There are N^c elementary components in each product that must be removed. The task of eliminating the component i is specified by its processing time T_i^{rm}, while a boolean flag of h_i indicates its hazardousness. The general problem is to assign all tasks to the workstations of the disassembly line to optimize the objective function. I make the following additional assumptions:

- There are N_a^{ws} workstations available that have been preliminarily prepared.
- All workstations are identical and capable of performing any component removal tasks.

- The cycle time is denoted by T^c. Each workstation should complete the assigned removal tasks for the current product in the disassembly line. The cycle time is preliminarily defined.
- A precedence graph describes the logical dependencies of component removal tasks. The vertices represent the components to be removed. The edges are directed, and there are two types: AND type and OR type edges. The removal process of a component i can be started only if all components from which a directed AND-type path goes to the vertex of i are already removed, and at least one of the components from which a directed OR-type path goes to the vertex of i is removed. Typically, the precedence graph is used in its transitive reduced form.
- A solution is described by a sequence of component removal tasks, where C_j denotes the jth component in the disassembly sequence.
- A workstation will perform a continuous range of component removal tasks.
- The total time for the workstation k to perform all its assigned tasks is denoted by T_k^{ws}.
- N_u^{ws} denotes the number of workstations used.

The main attributes of the disassembly line balancing problem are summarized in Table 3.2.

Table 3.2: Notations of disassembly line balancing problems

N^c	Number of disassembly tasks
N_a^{ws}	Number of available workstations
N_u^{ws}	Number of used workstations
T_i^{rm}	Part removal time of the component i
T^c	Cycle time
T_k^{ws}	Total time for workstation k to perform all assigned tasks
C_j	jth components in the disassembly sequence
h_i	boolean flag indicates the hazardousness of component i
d_i	demand of of the component i

3.2.2 Objective function

There are several ways to measure how well a disassembly line is balanced. Based on the different disassembly optimization solutions collected in Table 3.1, there are two main approaches to the objective: one is cost-benefit-based, and the other is based on the processed quantities. Following the mixed objective approach [440], I used a combination of three components in my analysis.

- $F_1 = \min\{\sum_{k=1}^{N_u^{ws}} (T^c - T_k^{ws})^2\}$ minimizes the total idle time of workstations used,
- $F_2 = \min\{\sum_{j=1}^{N^c} (j \cdot h_{C_j})\}$ forces to remove hazardous components as early as it can,
- $F_3 = \min\{\sum_{j=1}^{N^c} (j \cdot d_{C_j})\}$ supports removing components with higher demand earlier.

The preliminary defined circle time and the total sum of the component removal times and the number of used workstations determine the total idle time: $N_u^{ws} \cdot T_c - \sum_{i=1}^{N^c} T_i^{rm}$. The shorter idle time results in a higher processed quantity. The objective function F_1 amplifies the imbalance and contains the corresponding elements. Applying a quadratic term minimizes idle time and drastically decreases the imbalance. The objective functions F_2 and F_3 depend on the component's property and the disassembly sequence. The hazardousness of the component requires additional care or causes an extra risk, which motivates their removal as early as possible. A binary boolean indicator describes the hazardousness property. F_2 prefers to remove hazardous components earlier than non-hazardous ones. F_3 is similar to the construction F_2, except that demand values are not binary but non-negative figures, and they represent the importance of disassembled components in remanufacturing due to the benefit values. It was reviewed and shown [443] that using the multi-objective approach for disassembly optimization problems is quite general, and using them in the described format enables comparison of my results to external reference solutions. Further aspects can also be considered by adding other components to the objective to minimize the disassembly cost, maximize the profit obtained from the disassembly, or minimize environmental pollution.

In this chapter, according to the use cases analyzed, I will use a linear combination of the three selected objectives: $w_1F_1 + w_2F_2 + w_3F_3$. In this context, the weights of w_1, w_2, and w_3 play a double role. These should compensate for the scaling discrepancies of the objectives, and the weights can determine the relative importance of the objectives based on external preferences. The first role could be substituted for normalizing or standardizing the objectives. However, the second aspect cannot be replaced with an autonomous solution, although the relative importance changes constantly in real-world problems. A dynamic weighting optimization that reflects the external conditions is beyond the scope of my work. Therefore, I assume that the weights of w_1, w_2, and w_3 are preliminarily defined as external parameters.

3.2.3 Linear programming problem formulation

There are already formulations described for the disassembly problem in the literature [444], but I will present a new formulation with fewer decision variables. It stands for six significant types of decision variables:

- Type 1 decision variables describe which of the removable components is assigned to a disassembly sequence order number:

$$x_{i,j}^{Type1} = \begin{cases} 1 & \text{if component } i \text{ will be removed} \\ & \text{as } j\text{th task in the removal sequence} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

- Type 2 decision variables determine the process times of each step of the disassembly sequence:

$$x_j^{Type2} = T_{C_k}^{rm}. \tag{3.2}$$

- Type 3 decision variables describe the workstation on which the component will be removed:

$$x_{i,k}^{Type3} = \begin{cases} 1 & \text{if component } i \text{ will be removed on the workstation } k \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

- Type 4 decision variables determine whether a workstation will be in use or not:

$$x_k^{Type4} = \begin{cases} 1 & \text{if workstation } k \text{ will be in use} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

- Type 5 decision variables determine the total idle time of each workstation if it is in use:

$$x_k^{Type5} = T^c x_k^{Type4} \sum_i x_{i,k}^{Type3} C_i. \tag{3.5}$$

- Type 6 decision variables describe the workstation assignments in the disassembly sequence order:

$$x_j^{Type6} = \begin{cases} k & \text{if workstation } k \text{ will be assigned the } j\text{th component} \\ & \text{in the disassembly sequence} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Then constraints must be established to satisfy all the requirements collected in Section 3.2.1:

- Constraints guarantee that each component is listed exactly once in the removal sequence:

$$\sum_{j=1}^{N^{ws}} x_{i,j}^{Type1} = 1 \qquad \forall i \in \{1, \ldots, N^c\} \tag{3.7}$$

- The processing time of the workstation should not exceed the cycle time limit:

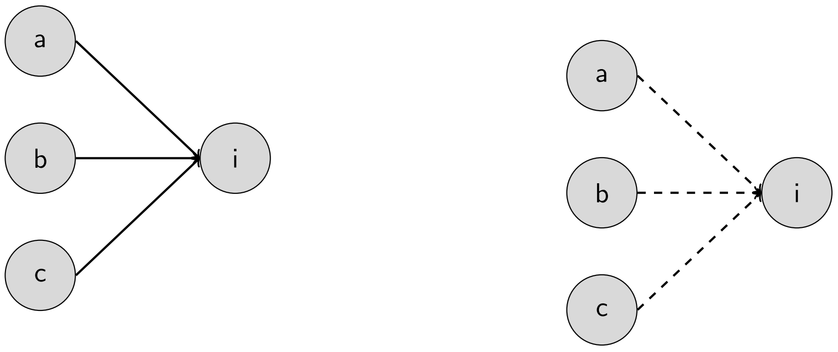
$$\sum_{i=1}^{N^c} T_i^{rm} \cdot x_{i,k}^{Type3} \leq T^c \qquad \forall k \in \{1, \ldots, N^{ws}\} \tag{3.8}$$

Before declaring the constraints of the precedence graph, the two types of predecessor relations need to be declared:

- The predecessor AND relation ($P_{AND}(i)$) declares a set of predecessor tasks that all need to be finished before starting task i.
- The predecessor OR relation ($P_{OR}(i)$) declares a set of predecessor tasks of which at least one needs to be completed before starting the task i.

Figure 3.2 shows predecessor AND and OR relations examples.

In this context, the definition of the required constraints can be continued:



(A) Predecessor AND relation (B) Predecessor OR relation

FIGURE 3.2: Predecessor relation types

- All the predecessor tasks with AND relation should be assigned earlier in the sequence than the particular one:

$$\sum_{j=1}^{N^c} j \cdot x_{l,j}^{Type1} \leq \sum_{j=1}^{N^c} j \cdot x_{i,j}^{Type1} \qquad \forall i \in \{1, \dots, N^c\}; \forall l \in P_{AND}(i) \tag{3.9}$$

- At least one of the predecessor tasks with OR relation should be assigned earlier in the sequence than the particular one:

$$x_{i,j}^{Type1} \leq \sum_{h=1}^{j} \sum_{l \in P_{OR}(i)} j \cdot x_{l,h}^{Type1} \qquad \forall i \in \{1, \dots, N^c\}; \forall j \in \{1, \dots, N^c\} \tag{3.10}$$

- Ensure that a disassembly task is assigned to exactly one workstation:

$$\sum_{k=1}^{N^{ws}} x_{i,k}^{Type3} = 1 \qquad \forall i \in \{1, \dots, N^c\} \tag{3.11}$$

- The workstation assignments should be in a monotone sequence:

$$0 \leq x_j^{Type6} - x_{j-1}^{Type6} \leq 1 \qquad \forall j \in \{2, \dots, N^c\} \tag{3.12}$$

- Integer and non-negative properties:

$$\begin{array}{ll}
x_{i,j}^{Type1} \in \{0,1\} & \forall i,j \in \{1,\dots,N^c\} \\
0 \leq x_j^{Type2} \leq T^c & \forall j \in \{1,\dots,N^c\} \\
x_{i,k}^{Type3} \in \{0,1\} & \forall i \in \{1,\dots,N^c\} \\
 & \forall k \in \{1,\dots,N^{ws}\} \\
0 \leq x_k^{Type4} \leq T^c & \forall k \in \{1,\dots,N^{ws}\} \\
x_k^{Type5} \in \{0,1\} & \forall k \in \{1,\dots,N^{ws}\} \\
x_j^{Type6} \in \{1,\dots,N^{ws}\} & \forall j \in \{1,\dots,N^c\}
\end{array} \tag{3.13}$$

By solving the MIQP problem, an optimal solution can be provided, but in practice, it can be a heavily resource-intensive process for mid- and large-scale problems. An earlier analysis of a profit-oriented linear objective shows that an exact MILP solution cannot be reached in a reasonable time limit for disassembly problems with more than 60 components [444]. In this formulation, the number of decision variables was quadratic to the number of components. I used a modified formulation with $2(N^c)^2 + 4N^c$ decision variables for the N^c-component disassembly problem. Moreover, the weighted objective leads to a quadratic optimization problem. I used a Gurobi-based Matlab solver and experienced the same issue: no solution was found within the same time limit[1]. This fact may bring alternative solutions to the fore, especially reinforcement learning methods.

3.3 Formulation of the disassembly line balancing problem as an RL-based optimization task

From the RL methods described in Section 2.2.1, I will focus on using an action-value function to determine the current optimal action. However, for large state- and/or action spaces, keeping the value function updated (and hence optimal) can be a very slow process.

Having briefly recalled, the update logic of the action-value function $q_\pi(s;a)$ can be determined by observing the results of a behaviour policy $\mu(a|s)$. Assume that in-state S_t, the very next action is derived using behaviour policy: $A_{t+1} \sim \mu(\cdot|S_t)$. Then the Q-learning value-function update will look like: $Q(S_t;A_t) \leftarrow Q(S_t;A_t) + \alpha\Big(R_{t+1} + \gamma Q(S_{t+1};A') - Q(S_t;A_t)\Big)$.

[1]Similarly to [444], I applied a time limit of 3.600 seconds for executions.

In a special case, if the target policy π is chosen as a pure greedy policy and the behaviour policy μ follows ϵ-greedy policy, then the so-called SARSAMAX update can be defined as follows: $Q(S;A) \leftarrow Q(S;A) + \alpha\Big(R + \gamma \max_{a'} Q(S';a') - Q(S;A)\Big)$. Last but not least, it was proven that the Q-learning control converges to the optimal action-value function: $Q(s;a) \to q_*(s;a)$.

3.3.1 Design of reinforcement learning solution

In this section, I will present a reinforcement learning-based solution design.

- *State space*: A state must contain all the relevant information from the past and be identical to those for equivalent situations. Therefore, the current state should contain the set of removed and remaining components, as well as the utilization of the active workstation. So, the state vector can be declared in a similar way to the multi-type decision variable in the MIQP formulation:

  - the performed removal steps and the remaining ones:

$$\hat{x}_i^{Type1} = \begin{cases} 1 & \text{if component } i \text{ has been removed} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

  - the current utilization of the active workstation:

$$\hat{x}^{Type2} = \text{total assigned removal time of last active workstation.} \tag{3.15}$$

- *Action space*: The next action is determined by considering both the current state and the constraints defined by the precedence graph. Formally, it will be described by the component's identifier, which one needs to be removed next. It is essential to highlight that a built-in heuristic to limit potential actions to the feasible ones can significantly speed up the solution.

- *Reward*: The reward function is defined as the reciprocal of the weighted sum of the objective components:

$$\frac{1}{w_1F_1 + w_2F_2 + w_3F_3}. \tag{3.16}$$

- *Reinforcement Learning method*: Considering that both state- and action spaces are discrete, it is obvious to use the Q-learning method. Triplets of the single state vector, the next action, and cumulative discounted rewards will determine Q-table rows (practically, Q-table structure).

- *Q-table growth*: Some approaches suggest declaring the Q-table structure initially, and during the learning phase, its rows need to be updated. Determining the total number of feasible states using a precedence graph is far from trivial. Therefore, a dynamic Q-table growth mechanism [445] was applied: the learning phase starts with an empty Q-table, and whenever a new state-action pair is observed, it should be inserted into the Q-table. Therefore, the Q-table contains only visited rows.

- *Knowledge transition*: Whenever the RL agent experiences a better reward from a visited state than the former best one, the Q-table must be updated accordingly. The learning process can be speeded up using the knowledge transition process. In this case, the Q-table updates are made backwards from the latter visited states of the episode to the former ones. The key idea is to update not only the visited state-action pairs of the episode but all further state-action pairs that lead to the visited route. In other words, the rewards of those state-action pairs, which partially overlap with the visited episode, can also be updated.

- If the agent should make the optimal action and not a random one, but it is not known (not listed in the Q-table yet) because the current state has never been visited before, then a random decision will be made as a fallback action.

- Disabling the discount factor can simplify the Q-learning method. The reason behind this is that the state defines how many actions are required to finish the episode, so the discount factor value could be easily calculated from the state. However, its ability to differentiate between potential routes by considering their lengths also breaks off.

- The Q-learning method works with ϵ-Greedy decisions: the RL agent takes a random action with ϵ probability or the known best action based on the Q-table with $(1 - \epsilon)$ probability. There are different ϵ-strategies, from which I tested the following four:

1. *Pure ϵ-Greedy approach*: during the whole simulation, the value of ϵ is constant in all the episodes.
2. *Two-step ϵ-Greedy approach*: in the first phase of the simulation ϵ has a value of 100%, and hence the RL agent takes only random actions, while in the second phase of the simulation ϵ switches to a lower reasonable constant.
3. *Linearly decreasing ϵ approach*: the value of ϵ starts from 100% at the beginning and linearly decreases to 0% proportionally to the progress of the simulation.
4. *Sigmoid-shape ϵ approach*: ϵ value goes from 100% to 0%, but in contrast to the linear version, it follows a sigmoid-shape curve.

Figure 3.3 shows the tested ϵ-functions by the progress of the simulation (in proportion to the scheduled episodes).

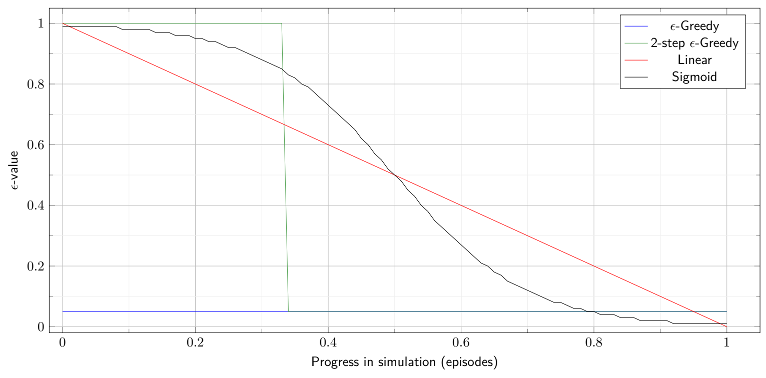


FIGURE 3.3: ϵ functions tested in ϵ strategies

3.3.2 Disassembly optimization algorithm with reinforcement learning (dlOptRL)

This section will describe my newly developed DLOPTRL algorithm. Its name encapsulates its functionality and method: a disassembly line optimization approach utilizing a reinforcement learning-based method combined with a built-in heuristic to determine the proper following action.

Generally, in reinforcement learning applications, all feedback arrives in the reward, meaning that the agent takes a sequence of actions and will experience

Algorithm 1 Procedure for disassembly line optimization with reinforcement learning

```
 1: function DLOPTRL($P$, $\mathbf{t}$, $\mathbf{h}$, $\mathbf{d}$, $c$, $\mathbf{o}$)
 2: Input:
 3:     $P$ $n \times n$ matrix represents the precedence graph  ▷ $n = |P|$: number of parts to remove
 4:     $\mathbf{t}$ $n$-length vector describes part removal times
 5:     $\mathbf{h}$ $n$-length vector indicates hazardousness of components
 6:     $\mathbf{d}$ $n$-length vector determines demand values
 7:     $c$ contains cycle time
 8:     $\mathbf{o}$ 3-length vector declares the objective components' weights
 9:                                                      ▷ initialize learning parameters
10:     $l \leftarrow 1500$                              ▷ learning length: number of episodes
11:     $Q((\mathbf{s_1}, s_2), a, r) \leftarrow [.]$    ▷ initialize Q-table
12:     for $i = 1$ to $l$ do                           ▷ execute $l$-episode simulation
13:         $(\mathbf{s_1}, s_2) \leftarrow \mathbf{0}$  ▷ reset state vector components:
14: ▷ $n$-length $\mathbf{s_1}$ describes the processed steps, scalar $s_2$ shows current workstation utilization
15:         $r \leftarrow 0$                             ▷ reset cumulative reward
16:         $k \leftarrow 1$                             ▷ set workstation assignment pointer
17:         $\mathbf{s} \leftarrow \mathbf{0}$           ▷ reset $n$-length component ordering vector
18:         $\mathbf{w} \leftarrow \mathbf{0}$           ▷ reset $n$-length workstation assignment vector
19:         for $j = 1$ to $n$ do
20:             $\mathbf{b} \leftarrow (\mathbf{1} - (((s_1 P_{AND}) < (\mathbf{1} P_{AND})).\cdot(s_1 P_{OR} == \mathbf{0}))).\cdot(\mathbf{1} - s_1)$   ▷ determine
    applicable steps
21:             $\xi_1 \leftarrow \backsim U(0,1)$     ▷ generate a random number by standard uniform distribution
22:             if $\xi_1 > \frac{i}{l}$ then           ▷ Q-table based optimal action
23:                 $m \leftarrow \max(r | q \in \{Q(\mathbf{s_1}, s_2, a, r) | \mathbf{s_1}, s_2\})$   ▷ get maximal expected reward
24:                 $\mathcal{M} \leftarrow \{(a | q \in \{Q(\mathbf{s_1}, s_2, a, r) | \mathbf{s_1}, s_2, r = m\})\}$   ▷ get maximal reward steps
25:                 $\mathbf{b} \leftarrow \mathbf{b}.\cdot \mathbf{a}_{\mathcal{M}}$   ▷ update applicable steps by intersecting maximal-reward steps
26:                 if $|\mathbf{b}| == 0$ then
27:                     goto else branch in line 29   ▷ if no applicable step, fallback to a random
28:                 end if
29:             else
30:                 $\xi_2 \leftarrow \backsim U(0,1)$   ▷ generate a random number by std. uniform distribution
31:                 $\mathbf{z}(x) \leftarrow \frac{1}{\|\mathbf{b}\|} \mathbf{b} U$   ▷ calculate cumulative distribution
32:                 $a \leftarrow \underset{x}{\operatorname{argmax}}\ \mathbf{z}(x) < \xi_2$   ▷ determine random action
33:             end if
34:             $\mathbf{o}(j) \leftarrow a$                ▷ store $j$th action in episode ordering vector
35:             $\mathbf{s_1}(a) \leftarrow 1$              ▷ register component $a$ as processed step
36:             if $c - s_2 > \mathbf{t}(a)$ then
37:                 $k \leftarrow k + 1, s_2 \leftarrow 0$   ▷ if workstation utilization requires, move ws. pointer on
38:             end if
39:             $\mathbf{w}(j) \leftarrow k$   ▷ assign workstation $k$ to the $j$th component removal action
40:             $s_2 \leftarrow s_2 + \mathbf{t}(a)$   ▷ add component $a$ removal time to workstation utilization
41:         end for
42:         $\mathbf{r} \leftarrow \mathit{reward}(\mathbf{h}, \mathbf{d}, c, \mathbf{o}, \mathbf{w})$   ▷ get rewards
43:         for $j = 1$ to $n - 1$ do
44:             $(\hat{\mathbf{s_1}}, \hat{s_2}) \leftarrow (\mathbf{s_1}, s_2)|_{\text{processed to step } j}$   ▷ get state vectors restricted until step $j$
45:             if $\exists(\hat{\mathbf{s_1}}, \hat{\mathbf{s_2}}, a, .) \in Q$ then   ▷ the state-action pair exists in the Q-table
46:                 if $r > \max_q Q((\hat{\mathbf{s_1}}, \hat{s_2}), \mathbf{s}(j+1), q)$ then   ▷ current reward > best in Q-table
47:                     $Q((\hat{\mathbf{s_1}}, \hat{s_2}), \mathbf{s}(j+1), q) \leftarrow Q((\hat{\mathbf{s_1}}, \hat{s_2}), \mathbf{s}(j+1), r)$   ▷ update Q-table record
48:                 end if
49:             else                                    ▷ no appropriate record in Q-table
50:                 $Q \overset{+}{\leftarrow} ((\mathbf{s_1}, s_2), a, r)$   ▷ insert Q-table record
51:             end if
52:         end for
53:     end for
54: end function
```

whether it works fine. In this approach, the agent is not restricted to preserving itself from an easily foreseeable bad action. Instead, it will realize the badness of the actions only afterwards by getting the low reward values. My algorithm implemented a heuristic within the action-choosing process to significantly decrease the potential action space's size. The principal idea was that all the restrictions declared in the MIQP formulation could be used to pre-filter possible actions before the agent chooses the final one. This approach helps the agent discover only the feasible part of the action space and not waste time exploring irrelevant paths, significantly speeding up the learning phase.

dlOptRL algorithm requires the following inputs:

- P_{AND} is an $n{\times}n$ matrix representing the predecessor AND relations of the precedence graph of the disassembly problem, where $n = |P|$ is the total number of parts to remove. According to the edge types of a precedence graph, the matrix elements have a binary applicable value set. The value of $P(i, j)$ is defined by the type of $e(v_i, v_j)$ as follows: 0 means that there are no direct predecessor AND dependencies between part i and j (represented by v_i and v_j on the precedence graph) during the disassembly process; 1 describes an existing predecessor AND relation between part i and j.
- P_{OR} is an $n{\times}n$ matrix representing the predecessor OR relations of the disassembly problem's precedence graph similar to the construction P_{AND}.
- $\mathbf{t}$ is an n-length vector that describes part removal times
- $\mathbf{h}$ is an n-length binary vector that indicates hazardousness of each component
- $\mathbf{d}$ is an n-length vector that determines the demand values of parts
- c is a constant that describes cycle-time
- $\mathbf{o}$ is a 3-length vector that contains the objective components' weights

After entering into an outer loop that iterates the episodes, some technical variables (counters and pointers) should be initialized, and an inner loop that represents a single episode should be started. First, the applicable steps should be determined by considering the current state of disassembly. Then, a decision should be made about how the next action will be determined. The agent will try to follow the known best option that will be successful in the case of a double condition being met, namely:

- the agent is on a known track, and hence at least one applicable step is known, and
- the generated standard uniform random number is over the uniformly decreasing threshold

If both of the above conditions are satisfied, the agent will exploit its cumulative knowledge and choose the optimal action that will result in the highest reward. Otherwise, the next action will be randomly selected from the applicable actions. The desired action will be registered in the short-term episode history.

After the episode ends, the agent will retrieve the reward. Then, it must check whether the visited state-action pairs are registered in the long-term Q-table. If an appropriate row is available in the Q-table, then the discounted reward value will be compared to the one stored in the Q-table. If the agent realizes that the newly experienced path provides a greater reward than the known best one, it must be updated. This is also a minor modification to the original Q-learning method, where the rows of the Q-table contain the average reward values. For a deterministic disassembly problem, the MAX aggregation function can be used, and it lets the agent immediately learn whenever a new best route is visited. Finally, if the observed state-action pair is not listed yet in the Q-table, then it should be added to the table.

3.3.3 Formulating dlOptRL algorithm as a HARL method

Since finding an optimal solution using RL methods can be very time-consuming, in recent years, many researchers have made efforts to speed up the learning process by improving the action selection method [446]. There are successful references for extracting domain knowledge by integrating special heuristics into an RL method [447]. It was shown [448] that the value function can be mathematically combined with a heuristic function if $\left[F_t(s_t, a_t) \bowtie \xi H_t(s_t, a_t)^{\beta}\right]$, where $\mathcal{F} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is an estimate of a value function, while $\mathcal{H} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the heuristic function. It can be easily seen that a sufficiently large negative value of ξ can practically avoid the selection of inappropriate actions. Regarding the dlOptRL algorithm, the heuristic function can be defined as 0 when the component is disassembled according to the precedence graph and $-\Omega$ otherwise, where Ω is greater than the theoretical maximum objective value of the concrete disassembly problem.

3.4 Application examples for disassembly line balancing problems

The literature analyzes several benchmark problems. In this section, I will present two of them and summarize the performance of my reinforcement learning-based solution by comparing it to classical methods. Then, I will highlight the main advantages of RL-based optimizations in further problems.

3.4.1 Small scale benchmark problem - Personal computer disassembly

There is a small-scale problem in the literature [440] [449] about disassembling personal computers. Eight salvageable components are identified in a PC. The parts themselves, their removal times, demand values and hazardousness indicators are collected in Table 3.3.

TABLE 3.3: Personal computer disassembly tasks and parameters

Task no.	Disassembly task	Removal time	Demand	Hazardousness
		[sec]	[$/1000 items]	[Y/N]
1	PC top cover	14	360	No
2	Floppy drive	10	500	No
3	Hard drive	12	620	No
4	Backplane	18	480	No
5	PCI cards	23	540	No
6	RAM modules (2)	16	750	No
7	Power supply	20	295	Yes
8	Motherboard	36	720	No

A precedence graph describes the logical dependencies of the order of the disassembly task in Figure 3.4.

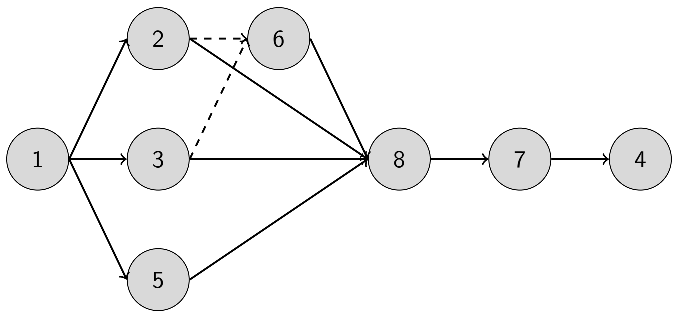


FIGURE 3.4: Precedence graph of personal computer disassembly problem

By choosing a combined objective function of $F = \frac{1}{w_1 F_1 + w_2 F_2 + w_3 F3}$, where the components are the same as defined in Section 3.2.2, the optimal global solution can be determined using an MIQP solver. Note that the weights allow prioritizing the objective components to align with the user's needs. Therefore, the concrete weighting values are less important from a scientific perspective, and hence, the researchers often set them equally. Although there are multiple reasons not to follow this, I also applied equal weights in this use case to let the results compare to existing benchmarks. Table 3.4 presents the published global optimum to remove the components.

TABLE 3.4: Optimal solution for personal computer disassembly problem

Sequence order	1	2	3	4	5	6	7	8
Component to remove	1	5	3	6	2	8	7	4
Part removal time [sec]	14	23	12	16	10	36	20	18
Assigned workstation	1	1	2	2	2	3	4	4
Workstation idle time [sec]		3			2	4		2

First, by solving the MIQP problem, I found that the optimal objective value is $F_{opt} = (3^2 + 2^2 + 4^2 + 2^2) + 7 + 19025 = 19065$. The result of DLOPTRL can be evaluated by comparing it with the known optimal objective as a reference value.

As a first approach, I implemented a simple Q-learning-based algorithm controlled purely by the reward function. I had only a single restriction in choosing the next part for removal. Every component needs to be selected precisely once in the disassembly sequence. I realized that this approach practically results in very low efficiency, such as testing a random permutation of parts to see whether it meets the criterion of the precedence graph or not.

This kind of experience gave me the motivation to integrate the constraints (that were collected in the MIQP formulation) into the next action determination step by restricting the set of potential actions only to the applicable ones, which is practically the intersection of three sets:

- parts that are not removed yet,
- parts of which all AND type predecessor parts are already removed,
- parts of which at least one OR type predecessor part is removed.

This limitation of the action space indicates a significant change in the learning speed: in the small-scale use case, the agent found the optimal solution after very few steps.

3.4.2 Mid scale benchmark problem - Cell phone disassembly

Another case study problem is about disassembling cell phones. Twenty-five salvageable components of a cell phone have been identified. The parts themselves, their removal times, demand values, and hazardousness indicators are collected in Table 3.5.

A precedence graph describes the logical dependencies of the disassembly order in Figure 3.5. This leads to a less trivial solution than in the previous use case [440]. This led me to use my new formulation that works with a significantly smaller vector size of decision variables as described in Section 3.2.3.

The solution to the degraded MIQP problem is presented in Table 3.6. In this case, the optimal objective value is: $F_{opt} = 15 + 75 + 815 = 905$. The pure Q-learning algorithm could not deliver a feasible solution without limiting the action space to the applicable actions. In contrast, the dlOptRL algorithm provides a feasible solution from the very first episode. Of course, this does not mean the early solutions are efficient enough. By performing multiple simulations, I found that the dlOptRL algorithm could find a reasonably good approximation for the optimal solution. As described in Section 3.3.1, I tested four different static ϵ-strategies for the RL agent's decision. Figure 3.6 shows the learning performance results following the different approaches.

Table 3.5: Cell phone disassembly tasks and parameters

Task no.	Disassembly task	Removal time [sec]	Demand [$/1000 items]	Hazardousness [Y/N]
1	Antenna	3	4	Yes
2	Battery	2	7	Yes
3	Antenna guide	3	1	No
4	Bolt (type 1) a	10	1	No
5	Bolt (type 1) b	10	1	No
6	Bolt (type 2) 1	15	1	No
7	Bolt (type 2) 2	15	1	No
8	Bolt (type 2) 3	15	1	No
9	Bolt (type 2) 4	15	1	No
10	Clip	2	2	No
11	Rubber seal	2	1	No
12	Speaker	2	4	Yes
13	White cable	2	1	No
14	Red/blue cable	2	1	No
15	Orange cable	2	1	No
16	Metal top	2	1	No
17	Front cover	2	2	No
18	Back cover	3	2	No
19	Circuit board	18	8	Yes
20	Plastic screen	5	1	No
21	Keyboard	1	4	No
22	Liquid crystal display	15	6	No
23	Sub-keyboard	15	7	Yes
24	Internal circuit	2	1	No
25	Microphone	2	4	Yes

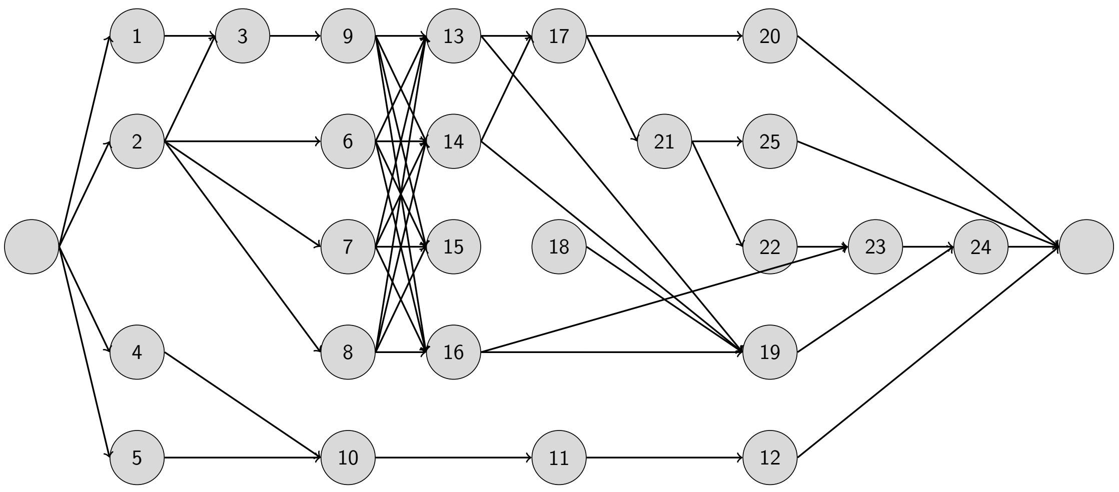


Figure 3.5: Precedence graph of cell phone disassembly problem

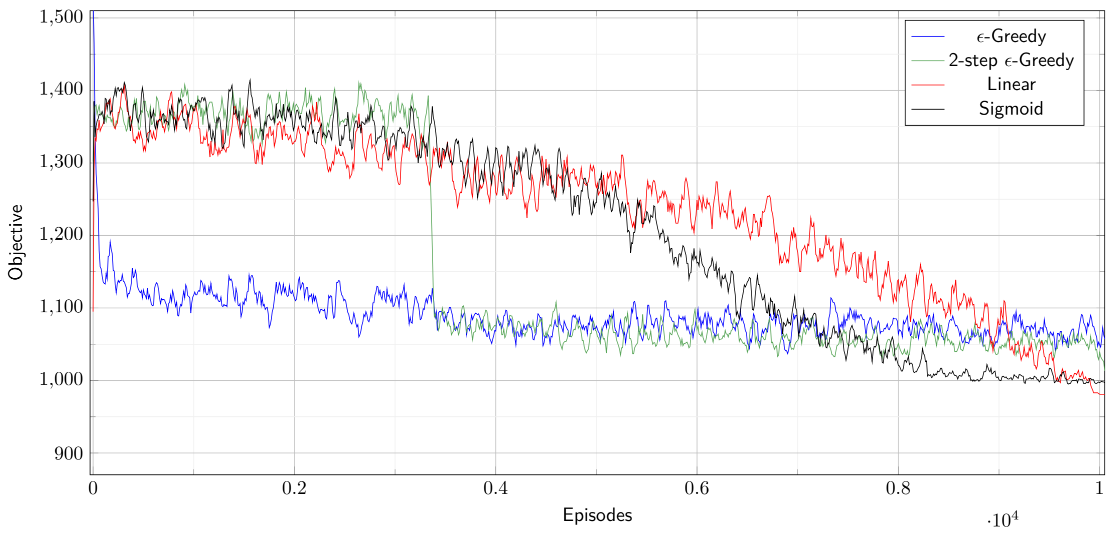


FIGURE 3.6: Learning performance of the tested ϵ-strategies

The weighted total objective values are plotted instead of cumulative rewards for a more straightforward interpretation. Although the continuous ϵ-Greedy approach presents the best performance with the lowest objective values in the first phase of the simulation, it reaches the worst solution at the end of the simulation. The other methods start with a high ϵ-value to discover the search space more intensively. The linearly decreasing ϵ approach gets the best overall objective value. Therefore, I will use it further. Another advantage is that it does not need a custom parameter for its operation, simplifying the DLOPTRL algorithm.

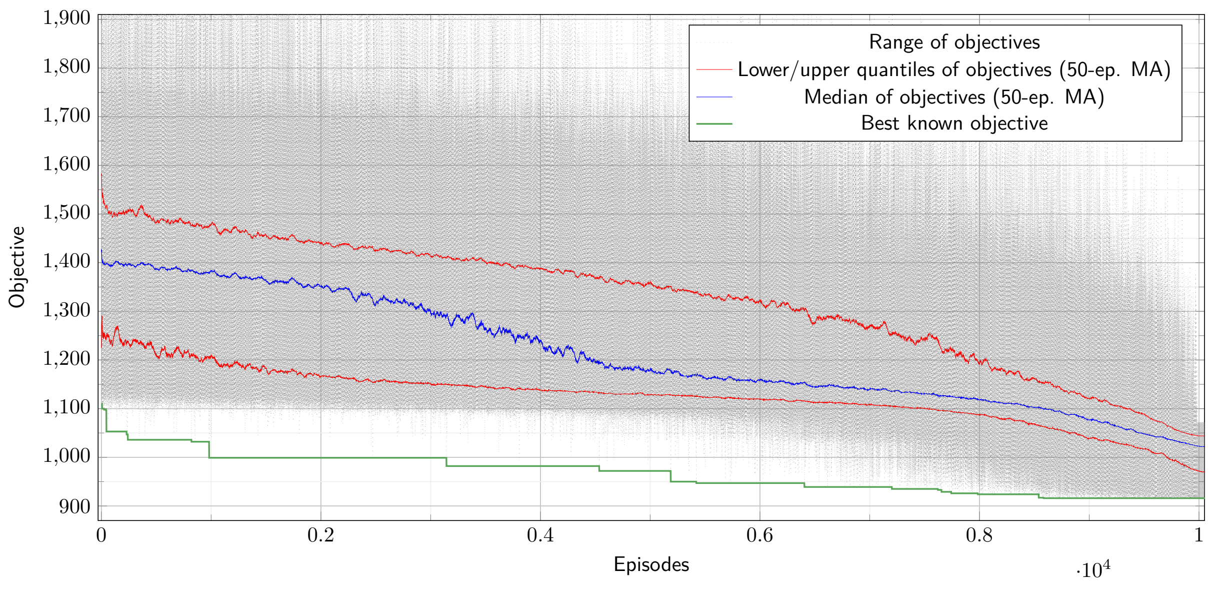


FIGURE 3.7: Objective values (reciprocal reward values) by iterations

In contrast to the small-scale use case presented in Section 3.4.1, the DLOPTRL algorithm does not reach the global optimal solution of the mid-scale benchmark problem. I executed 100 identically parameterized simulations to provide cross-validated results. Figure 3.7 shows the empirical results by presenting:

- the range of the observed objectives,
- 50-episode moving averages of median objectives,
- 50-episode moving averages of upper/lower quartiles of objectives,
- best objective of learned routes by episodes.

Although the global optimum was not found, all the simulations show a stable convergence in objective values. The median value of the objectives is 985, which is 8.8% worse than the global optimum. The best solution of the DLOPTRL method has an objective of 917 and is presented in detail in Table 3.7. A widely used indicator to compare two sequences is the concordance ratio. Calculated by counting all pairs of items in the same order in both sequences and dividing by the total number of pairs of items. Out of the $\binom{25}{2} = 300$ different item pairs, 293 have concordant orders, 7 have discordant orders, and the concordance ratio is 97.67% As the multiple steps of the best objective curve show, the learning process is continuous. There is a significant difference between the moving average values and the known best reward value, highlighting the "cost of learning". If the agent decides to explore an unknown path instead of the known best one, it will cause some loss in overall performance, but it leaves open the chance to find a better path than the current best one.

TABLE 3.6: Optimal solution for cell phone disassembly problem

Sequence order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Component to remove	2	7	1	8	3	6	9	14	13	17	21	22	25	16	15	23	18	19	20	4	5	10	11	12	24
Part removal time [sec]	2	15	3	15	3	15	15	2	2	2	1	5	2	2	2	15	3	18	5	10	10	2	2	2	2
Assigned workstation	1	1	2	2	3	3	4	4	5	5	5	5	5	5	5	6	6	7	8	8	9	9	9	9	9
Workstation idle time [sec]		1		0		0		1							2		0	0		3					0

TABLE 3.7: Best RL solution for cell phone disassembly problem

Sequence order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Component to remove	2	7	1	6	8	3	9	13	14	17	15	21	22	25	16	23	18	19	20	4	5	10	11	12	24
Part removal time [sec]	2	15	3	15	15	3	15	2	2	2	2	1	5	2	2	15	3	18	5	10	10	2	2	2	2
Assigned workstation	1	1	2	2	3	3	4	4	5	5	5	5	5	5	5	6	6	7	8	8	9	9	9	9	9
Workstation idle time [sec]		1		0		0		1							2		0	0		3					0

3.4.3 Comparison of RL-based solution to the mixed-integer solution and further research directions

In this section, I will summarize the central experiences of the two described solution methods for disassembly line balancing problems, as well as further research directions, which can have significant potential to improve the robustness and adaptivity of the solution.

As the disassembly line balancing problem is NP-hard [412], it cannot be guaranteed that an optimal solution will be found in the polynomial calculation time. In my formulation, the number of decision variables in the MIQP problem is quadratic to the number of components. I validated the formulation on the small-size PC disassembly problem, and the MIQP solver provided the optimal solution within a second. However, for the midsize cell phone disassembly problem, the MIQP solver processed 983,557 branches in 3,600 seconds. The first feasible solution was found after 539,992 iterations. In contrast, the RL-based solution reached a complete Q-table in 2 seconds for the small-size problem. Furthermore, it performed 10,000 episodes of the midsize problem in 382 seconds and grew a Q-table with 8,845 rows. Table 3.8 summarizes the key performance measures of the simulations.

Table 3.8: Summary of use case solutions

	PC problem		CP problem	
Solution type	MIQP	RL	MIQP	RL
Execution time [sec]	< 1	2	3,600	182
Final objective	19,065	19,065	1,269[1]	917[2]
Gap to global optimum	0	0	331	12
Relative gap to global optimum (%)	0%	0%	40.22%	1.33%

[1] Terminated because of the execution time limit.
[2] Median objective of 100 simulations.

The difference between the two solutions is that the MIQP solver requires a preliminary training process, whereas the RL-based solution is learned online. As my MIQP formulation shows, the component removal times, the hazardous indicators, the demands, the cycle time, and the objective component weights are all necessary to start the MIQP solver. In practice, it is often easier to set up an empirical reward function by measuring idle times and component removal orders and then letting the RL agent start learning. Furthermore, it is a complex task to implement an efficient MIQP solver, and it is costly to buy one, while my RL solution is easy to implement. In the case of multiple identical disassembly lines or if there is a virtual twin of them, the RL learning process is easy to parallelize. The results show that the RL agent reliably converges to the optimal solution. The dlOptRL algorithm delivers feasible solutions from the beginning and finds a competitive disassembly setup within a reasonable training time limit.

3.5 Summary of disassembly line balancing optimization

In this chapter, I showed that disassembly line optimization problems have become more important, leading researchers to devote more attention to developing dedicated solutions. Optimization challenges have many formulations, objectives, restrictions, and a wide range of problem sizes. I presented a compact formulation for the disassembly optimization problem that requires fewer decision variables to solve larger problems with the same solver limitations.

I showed that the standard approach of reinforcement learning application, when only the reward function must be declared, has a low convergence rate in the learning path. I described a Q-learning-based solution with an integrated heuristic named dlOptRl algorithm that lets the reinforcement learning agent learn the solution very effectively. I demonstrated the learning capability of my algorithm in two selected use cases that proved the real-life applicability of my approach.

The presented solution shows a possible way to fine-tune reinforcement learning algorithms to increase their learning performance for disassembly problems and other fields.

Furthermore, I have shown that my algorithm formally belongs to the Heuristically Accelerated Reinforcement Learning class. It delivers a working example of translating an MIQP problem into a heuristic function.

My algorithm has further potential for adapting to slowly changing disassembly environments or completely stochastic problems. The presented method can be used for other problem classes that need ordering complex actions into a sequence, such as the Traveling Salesman Problem (TSP), network/map discovery, or Vehicle Routing Problem (VRP), especially in stochastic cases where the state space is mixed continuous-discrete.

I identified further research directions that have significant potential to improve the solution's robustness and adaptivity.

- The dlOptRl algorithm can be extended to a multi-agent approach for parallelizing the learning process by updating a central Q-table.
- A new indicator for measuring the proportion of undiscovered routes (actions) would be worth introducing, and an adaptive episode length determination process could help better approach the global optimum.
- The removal times of the disassembly components have higher uncertainty than the assembly process times because the condition of a used product is more heterogeneous than a uniform new one. This implies applying stochastic removal times instead of deterministic ones.
- Using a moving time window or resetting the Q-table periodically can raise the adaptivity of the RL solution.
- The RL-based solution is less sensitive to measurement inconsistencies and one-time issues. Therefore, even if these observations are involved in the Q-value aggregations, their effects will be marginal in the long term.
- By directly measuring the rewards, the cumulative measurement errors can decrease compared to a MIQP formulation, where the errors will be accumulated.

The next phase of our ongoing research focuses on analyzing the behaviour of the RL-based solution in these contexts to verify them. I also recommend such an analysis for other researchers in the disassembly domain.

Chapter 4

Q-compression method

4.1 Introduction to constrained graph traversal problems

In this chapter, I focus on a special problem class that covers constrained graph traversal problems. Its interest comes from the property that its formulation leads to a mixed-integer (or mixed continuous-discrete) optimization problem, especially when distances are rather stochastic than deterministic. Although efficient solutions for the classical shortest pathfinding problems are known, these methods do not work with constraints and stochastically distributed distances. This motivated me to turn to the reinforcement learning method. In contrast with the most commonly used gradient-based solutions of mixed-integer optimization problems, I developed a new discretization approach for reducing the state space into a purely discrete space representation. I applied the iterative policy optimization method to that.

I identified a diverse set of problems belonging to the constrained graph traversal problem class. Although the shortest pathfinding problem is a well-known combinatorial optimization problem, its parameters are difficult to define precisely [450]. It is easy to see that the optimal route problem of a truck is practically a constrained shortest pathfinding problem [451], where the constraints describe the working hours limits of the driver and the availability of parking or the fuel capacity limits [452]. The daily route planning of an electric delivery van is a constrained Hamiltonian pathfinding problem, where the constraints are based on the limits of the battery capacity [453] and the charging options [454] or the battery exchange opportunities [455]. Furthermore, disassembling all product components after their life cycle is also a constrained graph traversal problem, where a precedence graph describes component removal dependencies and the constraints limit the use of parallel workstations [456]. Finally, I would like to highlight that in real-life problems, these kinds of problems are rather stochastic optimization tasks than discrete [457].

The classical methods for the shortest pathfinding problem, such as Dijkstra's [458], Floyd-Warshall [459], and Bellman-Ford algorithms, deliver the optimal solution in a reasonable time but are not able to handle stochastic distance distributions and complex constraints. The mixed-integer linear or quadratic programming technique can be an option in multiple cases because it supports the integration of constraints but is sensitive to stochastic parameters and the size of the problem [460]. The genetic algorithm demonstrated its power in stochastic search problems and converges to the Pareto-optimal solution. It not only provides the best solution but also alternatives. However, it requires built-in heuristics to retain cycles in the path, and its performance strongly depends on the proper parameterization setup [461]. In summary, there are no general methods to solve constrained stochastic graph traversal problems, so I propose using the reinforcement learning approach, which can handle both stochastic challenges and constraints well [462].

Reinforcement learning (RL) can be an obvious solution for sequential decision-making processes such as step-by-step pathfinding. The objective function should be implemented in the reward function, as well as in violation of the constraints [463]. However, additional difficulties arise when considering stochastic effects or measurement uncertainty: some continuous components must be integrated into the state space, which becomes mixed continuous-discrete. There are successful methods to solve the mixed-integer problem using gradient-based or deep RL methods [464]. RL techniques can also solve constrained combinatorial optimization problems [465]. Moreover, stochastic shortest-path problems also have RL solutions [466]. Compared to them, my research aims to discover a new algorithm by performing discretization steps using the DBSCAN clustering method and applying a discrete RL method for mixed continuous-discrete constrained stochastic graph traversal problems [R4].

4.2 Constrained graph traversal problem formulation

Due to the rapid development of technology and economics, optimization problems are increasingly focused on increasing efficiency. Numerous proven methods exist for finding optimal solutions for specific tasks, but there are cases where the classical methods cannot be applied directly or do not perform well. I present a special optimization problem class for which no general solution is known. I will formulate the problem and show that multiple real-world problems belong to the class.

Consider a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$. The vertex set is denoted by $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, while the edge set is described by $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ and finally the corresponding positive distances are in $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$. The function $d : \mathcal{E} \to \mathcal{D}$ can be used as an assignment to identify the corresponding distance of an edge: $d_i = d(e_i)$ for all $i \in 1, 2, \ldots, m$. The graph is assumed to be connected and simple in the context that there are no self-loops or multi-edges in it. The edge that connects the vertex i and the vertex j can be referred to as (v_i, v_j).

Suppose that the graph $\mathcal{G}$ has two privileged vertices: s and t are the starting and target vertex. A path is defined as a series of edges: $\mathcal{P} = (e_1, e_2, \ldots, e_k) = ((v_0, v_1), (v_1, v_2), \ldots (v_{k-1}, v_k) \subset \mathcal{E}$. The shortest pathfinding problem (SPP) is to find a path from s to t with the shortest total distance: $\min |\mathcal{P}| = \min \sum_{i=1}^{k} d_i$, where $v_0 = s$, $v_k = t$ and d_i is the distance value of the edge $e_i = (v_{i-1}, v_i)$.

Further limitations can be added to the shortest pathfinding problem: assume that the traveller can cover only a limited distance in one go. Therefore, the shortest path should be divided into subroutes or ranges that do not exceed their distance limit L:

$$\mathcal{P} = \cup_{i=1}^{l} \mathcal{P}_i \tag{4.1}$$

$$\sum_{i \in \mathcal{P}_j} d_i \leq L \text{ for all } j \in \{1, 2, \ldots, l\} \tag{4.2}$$

In the formulation, index i goes on the edges of the shortest path, while index j goes on the ranges. Equation 4.1 declares that the disjoint union of the ranges forms the shortest paths. Equation 4.2 limits the range length for all ranges, of which the number is assumed to be l. It is easy to see that the total length of the shortest path is equal to the total length of the ranges or the sum of the length of the involved edges: $|\mathcal{P}| = \sum_{j=1}^{l} |\mathcal{P}_j| = \sum_{i=1}^{k} d_i$.

An important relevance of a constrained shortest path problem in practice is the truck routing problem. In this case, the traveller should split his route into feasible ranges due to the driver's working hours limits and parking availability or fuel capacity limits. A secondary goal is to minimize the number of ranges in the whole path. To reach this, it is mandatory to extend the objective of the classical shortest pathfinding problem and declare the common scale for it, which can be the cost:

$$\min z = \big(c_d \sum_{j=1}^{l} |\mathcal{P}_j| + c_r l\big) \tag{4.3}$$

, where c_d represents the distance proportional cost of the completed path and c_r covers the range cost (e.g. battery recharge cost). This type of problem will be called the constrained shortest pathfinding problem (CSPP).

Other problems can be transformed into a CSPP. Consider a connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The Hamiltonian pathfinding problem is to find a continuous path in the graph that visits all the vertices exactly once. The problem can be reduced to a shortest pathfinding problem. A reversible transformation $\hat{t}$ can be performed on graph $\mathcal{G}$ to prepare graph $\hat{\mathcal{G}}$ as follows:

- Every possible loop-free route $v \subseteq \mathcal{V}$ declares a vertex $\hat{v}$ in the transformed graph, including $\hat{v}_0 = ()$ which represents the empty subset.
- $\hat{e}_{ij} = (\hat{v}_i, \hat{v}_j)$ is an edge in the transformed graph if and only if:
  - The path in $\mathcal{G}$ represented by $\hat{v}_i \subset \mathcal{V}$ is a sub-path of the one represented by $\hat{v}_j \subset \mathcal{V}$:
  $$\hat{v}_i \subset \hat{v}_j$$
  - The path of $\hat{v}_j$ is longer exactly by one edge than the path of $\hat{v}_i$:
  $$\hat{v}_i \cup (v_y) = \hat{v}_j, \text{ where } v_y \in \mathcal{V}$$
  - By marking with $v_x \in \mathcal{V}$ the last element of $\hat{v}_i$, the additional edge of path $\hat{v}_j$ is at its end compared to the path of $\hat{v}_j$:
  $$(v_x, v_y) \in \mathcal{E}$$

  - The initial vertex can be freely chosen from all vertex $v \in \mathcal{V}$

$$((), (v)) \in \hat{\mathcal{E}} \text{ for all } v \in \mathcal{V}$$

- As in my formulation of the classical Hamiltonian pathfinding problem, there were no distances in the original graph $\mathcal{G}$. However, the SPP formulation requires distances from the edges, so a constant value 1 can be assigned to all existing $(\hat{v}_i, \hat{v}_j)$ edges of the transformed graph $\hat{\mathcal{G}}$. However, in real-world problems, distances are given: $\hat{d}(\hat{v}_i, \hat{v}_j) = d(v_x, v_y)$ and I am interested in solving the shortest Hamiltonian pathfinding problem (SHPP).
- Finally, an optimal solution to the shortest pathfinding problem of $\hat{\mathcal{G}}$ will provide the shortest Hamiltonian path with an objective function of $\min |\mathcal{H}| = \min \sum_{i=1}^{k} d_i$.

The graph $\hat{\mathcal{G}}$ can be simplified by merging vertex pairs $\hat{v}_a$ and $\hat{v}_b$ into a single transformed vertex if both contain the same original vertices and their last elements are identical. (For example, (v_1, v_2, v_3) can be merged into (v_2, v_1, v_3), but cannot be merged into (v_3, v_2, v_1)). The merging concept comes from the observation that it does not matter in a route what the vertex visiting order was, only the fact whether a vertex was visited or not, and the last visited vertex, which determines where the traveller can move on. Denote by $\hat{v}_{\max} \subset \hat{V}$ those transformed vertices that contain all $v \in \mathcal{V}$. Then it can be shown that finding a Hamiltonian route in graph $\mathcal{G}$ indicates a path from $\hat{v}_0$ to $\hat{v}_{\max}$ in $\hat{\mathcal{G}}$, and vice versa.

Similarly to CSPP, another constraint can be introduced to limit the distance the traveller can cover in one go. Formally, the Hamiltonian path $\mathcal{H}$ should be split into subroutes or ranges that do not exceed their distance limit L: $\mathcal{H} = \cup_{i=1}^{l} \mathcal{H}_i$, where $\sum_{i \in \mathcal{H}_|} d_i \leq L$ for all $j \in \{1, 2, \ldots, l\}$. It is easy to see that $|\mathcal{H}| = \sum_{j=1}^{l} |\mathcal{H}_j|$.

The constrained shortest Hamiltonian pathfinding problem (CSHPP) objective will contain two terms. The first one minimizes the total weighted route of the traveller, while the second term enforces performing the whole route with the smallest number of required ranges:

$$\min z = \big(c_d \sum_{j=1}^{l} |\mathcal{H}_j| + c_r l\big) \tag{4.4}$$

, where c_d represents the proportional cost of the distance of the completed path and c_r covers the range cost.

A relevant practical problem is the daily route planning of an electric delivery van. The van should visit all target addresses, considering the limits of the battery capacity and the charging options or battery exchange opportunities.

Another type of problem that can be transformed into a CSPP is the disassembly line balancing problem. The problem was described in Section 3.2.1 in detail. In its simplest form, I consider a disassembly line for a single product with a finite supply. Each product has n elementary components to remove, represented by a vertex set $\mathcal{V}$. The task of eliminating the component $v_i \in \mathcal{V}$ is specified by its processing time $t_i \in \mathcal{T}$, while a boolean flag of $h_i \in \mathcal{H}$ indicates its hazardousness, and finally $d_i \in \mathcal{D}$ declares its demand value. The general problem is determining the order of disassembly of the components and assigning every task to the workstations of the disassembly line to optimize the objective function. The predefined cycle time is denoted by $t_c \in \mathbf{R}$, and each workstation should complete its assigned removal tasks within the cycle time. A directed precedence graph $\mathcal{P} = (\mathcal{V}, \mathcal{E}, (\mathcal{T}, \mathcal{H}, \mathcal{D}))$ describes the logical dependencies of component removal tasks. As I mentioned above, the vertices represent the components to be removed. A component v_i can be removed only if all components from which a directed edge goes to the vertex v_i have already been removed. Typically, the precedence graph is used in its transitive reduced form. Consider a feasible solution that declares the order of component removals: the component represented by the vertex v_i should be removed as the r_ith element, and w_i^j determines which component should be removed on the workstation i as the jth task. Then, the objective function can be formulated as follows:

$$\min\Big(c_i \sum_{i=1}^{l} (t_c - \sum_{j=1}^{l_j} t_{w_i^j})^2 + c_h \sum_{i=1}^{n} h_i r_i + c_{de} \sum_{i=1}^{n} d_i r_i\Big) \tag{4.5}$$

The first term determines the quadratic idle times of the used workstations, which supports minimizing idle times and decreasing imbalance. The second and third terms depend on the component property and disassembly sequence and aim to remove the hazardous components and the components with higher values earlier. The three terms are weighted by the idle cost of c_i, the hazardous cost c_h, and the demand cost of c_{de}, which have multiple goals behind them: compensating for the scaling discrepancies of the objectives and determining the relative importance of the objective components based on external preferences. The cost of idle time c_i

integrates the distance proportional cost c_d, and the range cost c_r used in Equation 4.3 and Equation 4.4.

It is easy to see that the disassembly line balancing problem is a special modified case of a constrained Hamiltonian pathfinding problem since all the components should be removed. However, on the one hand, the disassembly of the components depends not only on the last removed components but also on the previously removed components. On the other hand, the objective function is more complex, with a quadratic term in it. Similarly to the Hamiltonian pathfinding problem, a reversible transformation $\tilde{t}$ can be constructed to formulate a problem as a CSPP to produce graph $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}}, (\tilde{\mathcal{T}}, \tilde{\mathcal{H}}, \tilde{\mathcal{D}}))$, for which the objective function becomes to the following form:

$$\min z = \big(c_i \sum_{j=1}^{l} (t_c - |\tilde{\mathcal{H}_j}|)^2 + c_h \sum_{j=1}^{n} \tilde{h_j j} + c_{de} \sum_{j=1}^{n} \tilde{d_j j}\big) \tag{4.6}$$

The problems described in the real world are rather stochastic than deterministic. In routing problems, distances are measured on a time scale that depends on traffic, weather, and other external conditions, while in disassembly problems, the component removal times depend on product conditions.

In the formulation of stochastic shortest pathfinding problems, the directed graph has the same structure as in the deterministic case, so the vertex set and the edge set are identical, but the related distances are probability variables: $\mathbf{D}_i \sim \text{LogNormal}(\mu_{d_i}, \sigma^2_{d_i})$ for all $i = 1, \ldots, m$.

4.3 Q-table compression method

This section presents a discretization method integrated into the Q-table-based policy iteration algorithm to solve mixed continuous-discrete problems with a reinforcement learning approach. First, I recall the action-value-based policy iteration methods, especially the every-visit Monte Carlo method and the Q-learning concept. Then, I declare the basic requirements for a state compression solution. Finally, I describe the Q-table compression process in detail by providing the pseudo-algorithm to support its implementation.

4.3.1 Reinforcement learning method for graph traversal problems

As I presented in Section 2.2, Reinforcement learning solves problems due to sequential learning: the agent observes the travelled path, which is a sequence of the visited vertices and needs to decide on the next action that should define the next vertex to visit. Therefore, RL techniques can provide an obvious solution for graph traversal problems by determining a sequence of vertices to find an optimal (shortest) path.

For a graph traversal problem, the action space is the vertex set: $A_t \in \mathcal{V}$, the observation space is the subroute performed: $O_t = ((v_0, v_1), (v_1, v_2), \dots (v_{t-1}, v_t) \subset \mathcal{E}$, while the reward is the increase in the partial objectives: $R_t \in \mathcal{R} = z_t - z_{t-1}$

The state value function $v(s)$ gives the expected total discounted return if starting from state s: $v(s) = \mathbb{E}[G_t \mid S_t = s]$. The Bellman equation practically states that the state value function can be decomposed into two parts: immediate reward (R_{t+1}) and discounted value of successor states $\gamma v(S_{t+1})$.

The policy covers the agent's behaviour in all possible cases, so it is essentially a map of states to actions. There are two main categories in it: deterministic policy $(a = \pi(s))$ and stochastic policy $(\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s])$.

I will use an action-value function to determine the current optimal action. However, it can be a very slow process for large state- and/or action spaces to keep the value function updated (and hence optimal). Denote by $N(s, a)$ the counter of visiting the state s with an action selection of a. Then consider the function $Q(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$, which accumulates the expected total discounted reward starting from state s and chooses action a. According to the every-visit Monte Carlo policy evaluation process, if the agent performs a new episode and receives rewards accordingly, then the counter must be incremented: $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$, and the Q-value function must be updated for all visited (S_t, A_t) pairs: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\big(G_t - Q(S_t, A_t)\big)$. It is proven that if the agent follows the update of the Q value function combined with a simple ϵ-Greedy action selection method, then the Q-value function approaches the optimal policy as the number of observations tends to infinity.

The Q-learning value-function update will look like the same as described in Section 3.3: $Q(S_t; A_t) \leftarrow Q(S_t; A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}; A') - Q(S_t; A_t)\big)$.

My solution used the every-visit Monte Carlo method to determine the optimal policy. However, this can be easily replaced by Q-learning, but in that case, it is necessary to choose a suitable α-strategy.

4.3.2 Q-table compression process

In this section, I collect the basic requirements of a discretization function, then present the Q-table compression process in detail, and highlight some hints and guidelines for implementation.

In general, Q-table methods are used for discrete problems. However, there are cases where the discrete problems partially become continuous by considering some stochastic effect or measurement uncertainty. One option to decrease the complexity of such a problem is to divide it into sub-problems. It can be done by introducing sub-goals and solving these problems separately [467]. Another option is to merge those observed states into a common one that differs from each other only insignificantly. This can be done by introducing a grid representation in the continuous observation space [468]. This section presents an integrated method for dynamically redefining the state space and transforming the Q-table to align it to the changes. I assume a stochastic Markov Decision Process with a mixed-integer state space and discrete action space.

o^i_j denotes the jth observation in episode i, and it is an element of the observation space O. The goal is to define a discrete representation S that will be used as a state space in the reinforcement learning process. Formally, I am looking for an iteratively refined mapping function $\mathcal{F}^i : O \to S$.

I can declare the following requirements for the functions $\mathcal{F}^i$.

- Assume that the optimal action is determined for each and every o^i_j, and it is denoted by a^i_j. If $a^i_j \neq a^k_j$ then $\mathcal{F}(a^i_j) \neq \mathcal{F}(a^k_j)$. This practically means that the mapping function only merges observation states if their optimal actions do not differ.

- The mapping function $\mathcal{F}$ should be consistent in that sense that if action a^i_j moves the agent from the observation state o^i_j to o^i_{j+1} then action a^i_j should move the agent from $\mathcal{F}(o^i_j)$ to $\mathcal{F}(o^i_{j+1})$.

- $|S| \ll |O|$, or in other words, the size of the state representation S should be significantly smaller than the observation space O. The smaller S is the better representation.

The first requirement prevents merging states that have different optimal actions. The second defines the transitivity of the mapping function. The last requirement states that the potential for using the described representation function depends on the efficiency of dimension reduction. If no significant reduction can be reached, then the method cannot increase the efficiency of the learning process.

I designed a modified Q-table-based policy iteration method according to the above requirements by integrating a discretization step. In the most straightforward RL framework, the agent observes its states directly and takes its actions accordingly. In my approach, the agent takes observations, determines the simplified state, and takes actions based on it. So observations determine the state, but observation space differs from state space: the first one is a mixed continuous-discrete space, while the last one is purely discrete. The key component in the process is the state space definition and its projection.

I suggest maintaining a compression-based Q-table learning process due to the following steps, which are summarized in Figure 4.1:

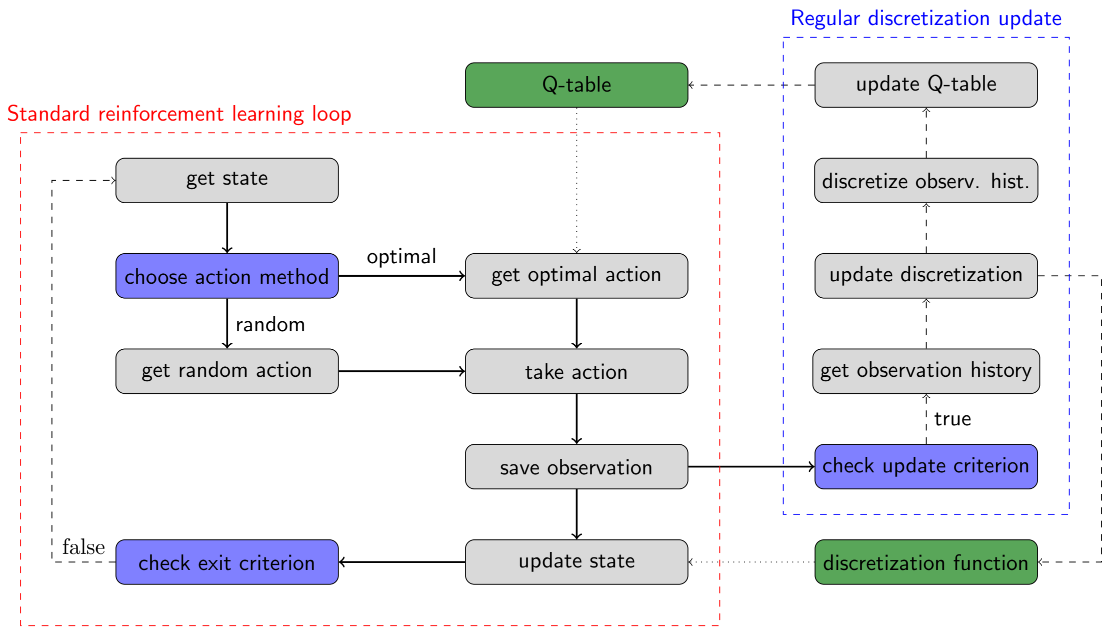


Figure 4.1: Q-compression process flow

There is a standard reinforcement learning loop in the process. The agent queries its current discretized state (`get state`). The learning process is based on the ϵ-Greedy algorithm: the agent decides whether to take a random or best-known action (`choose action method`). The agent randomly chooses an action with probability ϵ of the feasible actions (`get random action`) or chooses the optimal action from the feasible actions with $(1 - \epsilon)$ probability (`get optimal action`). In this case, the agent determines the expected cumulative rewards to reach the target state for all feasible actions first and then chooses the action with the best total expected reward (or if there are multiple actions with the same total expected reward, then randomly selects one out of them). If the Q-table does not contain a relevant entry for the current state because the trajectory is undiscovered, then the action selection falls back to the random method. The parameter value of ϵ decreases from 1 to 0 linearly as the number of episodes approaches its predefined limit. It should be highlighted that the restriction on potential actions improves the efficiency of the learning process compared to enabling any action and producing a bad reward for unfeasible actions.

Once the action is chosen, the agent performs the selected action, determines the new observation state, and registers the reward (`take action`). After that, the agent saves the quadlet (old state, action, new state, reward) into the observation history (`save observation`) and determines the new discretized state by applying the discretization function for the observation (`update state`). Finally, the agent checks whether the target state has been reached (`check exit criterion`). If not, then the cycle repeats.

Next to the standard reinforcement learning loop, the agent needs to update the discretization regularly and, on top of that, compress the state space size. The discretization update process is quite resource-intensive. Therefore, it is suggested that it be performed after a batch of episodes. Once the update criterion is satisfied (`check update criterion`), the agent queries the observation history (`get observation history`). To let the agent act adaptively, the history can be accessible due to a moving observation window to get the most relevant part of the history and not process the old, invalid records from there. The primary goal at this stage is to create a new/updated discretization ruleset that fits the requirements described above in this subsection and projects the mixed continuous-discrete observation space to a discrete state space (`update discretization`). Denote by $s_t = (v_t, u_t) \in \mathcal{V} \times \mathbf{R}$ an observation, where v_t shows the currently visited vertex of the graph and u_t describes the current range utilization. The agent takes the observations in a loop, fixes the discrete part of the observation space, and queries all matching records: $(v_i, u_i)|v_i = v_{fixed}$. Then, it uses the DBSCAN algorithm to split the continuous part of the observations into ranges. The number of ranges is finite and somewhat limited, but definitely discrete: $\bigcup_{i=1}^{j_t} R_i = \mathbf{R}$ and $\bigcap_{i=1}^{j_t} R_i = \emptyset$.

Multiple clustering algorithms were empirically tested to find the most suitable method for determining the optimal number of ranges and optimizing their hyperparameters. Hierarchical clustering with dendrograms, the k-means elbow method, the Bayesian information criterion, and the Akaike information criterion methods were also examined, but DBSCAN achieved the most stable partitioning. Figure 4.2 demonstrates the result of applying the DBSCAN algorithm to determine clusters in two selected discretization steps. It is observable that the clusters effectively separate the density peaks of the range utilization values.

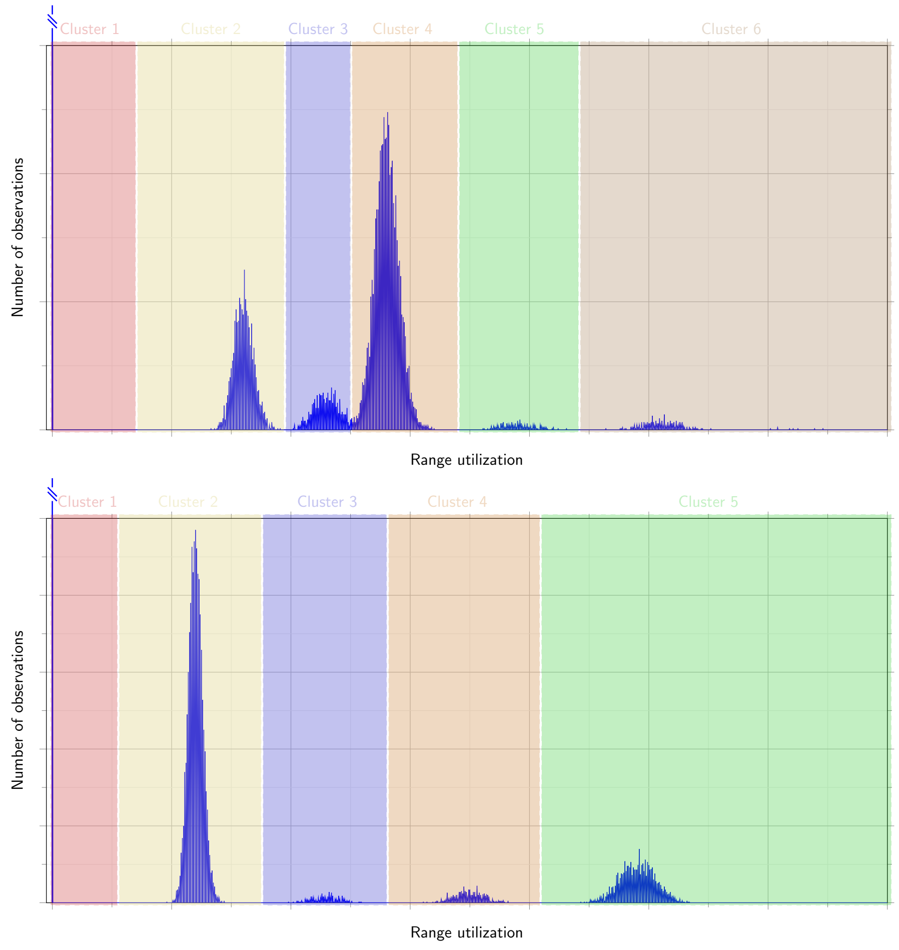


Figure 4.2: Demonstration of applying DBSCAN for clustering range utilization for two selected discretization steps

The agent repeats the discretization process for all elements in the loop. After updating the discretization function, the agent applies it to the observation history to produce a discretized version of it: $\mathcal{F}_{\rangle} : \mathcal{V} \times \mathbf{R} \to \mathcal{V} \times \{R_1, R_2, \dots R_{j_t}\}$ (`discretize observation history`). Finally, the agent truncates and rebuilds the Q-table from the discretized observation history using the discretized states (`update Q-table`). The algorithm 2 presents the Q-compression method in pseudocode format.

Algorithm 2 Q-table compression reinforcement learning method

```
1: function Q-Compression($D$, $l_{sim}$, $l_{range}$)
2: inputs:
3:     $D$: $n \times n$ matrix                         ▷ represents the graph distances
4:     $l_{sim}$: constant parameter                    ▷ define simulation length
5:     $l_{range}$: constant parameter                  ▷ define single range limit
6:                                                      ▷ initialize learning parameters
7:     $Q() \leftarrow [.]$                             ▷ initialize Q-table
8:     $O() \leftarrow [.]$                             ▷ initialize observation history
9:     for $i = 1, 2, \dots, l_{sim}$ do                ▷ loop for iterating episodes
10:        $(v, u)_{curr} \leftarrow (v_{start}, 0)$    ▷ reset state (current vertex, range util.)
11:        while $v_{curr} \neq v_{target}$ do          ▷ check episode exit criterion
12:            $\xi \leftarrow \backsim U(0, 1)$        ▷ generate standard uniform random number
13:            $\mathbf{a} \leftarrow \{v | D(v_{curr}, v) > 0\}$   ▷ get feasible action options list
14:            if $\xi < \frac{i}{l_{sim}}$ then        ▷ optimal action selection criterion
15:                $m \leftarrow \max\big(r = Q(s, a) | s = (v_{curr}, \mathcal{F}(u))\}\big)$   ▷ get maximal
   expected cumulative reward from Q-table
16:                $\mathbf{v_{max}} \leftarrow \big\{a | Q(s, a) = m, s = (v_{curr}, \mathcal{F}(u))\big\}$   ▷ get all
   maximal-reward actions from Q-table
17:                if $|\mathbf{v_{max} > 0}|$ then     ▷ If no applicable action found then
   fallback to random action
18:                    $\mathbf{a} \leftarrow \mathbf{a} \cap \mathbf{v_{max}}$   ▷ restrict optimal action list to optimals
19:                end if
20:            end if
21:            $a \leftarrow \backsim U(\mathbf{a})$    ▷ choose next action from options
22:            $v_{next} \leftarrow a, d \backsim D(s_{curr}, a), u_{next} \leftarrow u - d$   ▷ determine next obs.
23:            $\mathbf{r} \leftarrow$ Reward$(s_{curr}, a, d)$   ▷ get reward
24:            $O \overset{+}{\leftarrow} ((s_{curr}, a, s_{next}, r)$   ▷ save observation into history
25:            $v_{curr} \leftarrow v_{next}, u_{curr} \leftarrow u_{next}$   ▷ update state
26:        end while
27:        if $\bmod (i, l_{range}) == 0$ then          ▷ Q-table update due criterion
28:            Update-Q$(O)$                            ▷ call discret. and Q-table update sub-process
29:        end if
30:    end for
31: end function
```

Algorithm 1 Q-table compression reinforcement learning method cont.

```
function REWARD($s_{curr}$, $a$, $d$, $l_{range}$)
inputs:
    $s_{curr}$: pair of current vertex and range utilization
    $a$: single vertex                    ▷ action $a$ determines the next vertex to visit
    $d$: dynamic value          ▷ measures distance realized on performed action $a$
    $l_{range}$: constant parameter                             ▷ define utilization limit
    $r_d \leftarrow c_d \cdot d$                    ▷ get reward term of distance proportional cost
    $r_r \leftarrow c_r(d == u_{curr})$               ▷ get reward term of range proportional cost
    $r_o \leftarrow c_o(l_{range} < u_{curr})$           ▷ get reward term of range overutilization cost
    return $r_d + r_r + r_o$
end function

function UPDATE-Q($O$)
inputs:
    $O$: list of quad-tuples        ▷ stores the observation history up to episode $i$
    for $v \in \mathcal{V}$ do                                 ▷ loop for iterating vertices
        $O_v \leftarrow \{O(s_{curr}, a, s_{next}, r) | s_{curr} = (v, .)\}$          ▷ filter for relevant obs.
        $\mathcal{F}(u|v) \leftarrow \text{AVERAGE}\big\{\text{DBSCAN}\big(O_v(r)\big)\big\}$ ▷ calculate expected reward for the intervals discretized by DBSCAN
    end for
    $Q() \leftarrow [.]$                                                   ▷ reset Q-table
    for $j = 1, 2, \ldots, |O|$ do                          ▷ loop for iterating obs. history
        $\big((v, u)_{curr}, a, r, (v, u)_{next}\big) \leftarrow O_j(s_{curr}, a, r, s_{next})$          ▷ read $j$th obs.
        $\tilde{s}_{curr} \leftarrow \big(v_{curr}, \mathcal{F}(u_{curr})\big)$                ▷ determine discretized current state
        $\tilde{s}_{next} \leftarrow \big(v_{next}, \mathcal{F}(u_{next})\big)$                  ▷ determine discretized next state
        $Q(\tilde{s}_{curr}, a) \xleftarrow{+} \frac{1}{|O(\tilde{s}_{curr}, a, ., .)|}\big(r + \gamma \max_{a_{next}} Q(\tilde{s}_{next}, a_{next}) - Q(\tilde{s}_{curr}, a)\big)$ ▷ calculate expected reward for the intervals discretized by DBSCAN
    end for
end function
```

4.4 Results and Discussion

In this section, I present three application examples that demonstrate the usability of the Q-compression method. The first use case is a constrained shortest pathfinding problem (CSPP), the second is a constrained shortest Hamiltonian pathfinding problem (CSHPP), and the last is a disassembly line balancing problem (DLBP). The performance of the Q compression method is presented and compared with a simple grid-based discretization method and the optimal solution of the deterministic version of the problems.

If it is assumed that the constrained graph traversal process is a sequence of decisions to determine which vertex should be visited next, then two pieces of information influence the decision: the currently visited vertex and the range utilization. In Table 4.1, I summarize the key components of the class of constrained graph traversal problem.

Table 4.1: Constrained graph traversal problem components by problem types

Vertex setup (state space)	
CSPP	current vertex
CSHPP	visited vertices + current vertex
DLBP	removed components
Edge context (action space)	
CSPP	integrates the vertex connectivity
CSHPP	integrates the vertex connectivity
DLBP	integrates the precedence graph
Constraints (restrictions for action selection)	
CSPP	range utilization $\leq$ battery capacity
CSHPP	range utilization $\leq$ battery capacity
DLBP	workstation utilization $\leq$ cycle time
Objective	
CSPP	$\min\left(c_d \sum_{j=1}^{l} \lvert\mathcal{P}_j\rvert + c_r l\right)$
CSHPP	$\min\left(c_d \sum_{j=1}^{l} \lvert\mathcal{H}_j\rvert + c_r l\right)$
DLBP	$\min\left(c_i \sum_{i=1}^{l}(t_c - \sum_{j=1}^{l_j} t_{w_i^j})^2 + c_h \sum_{i=1}^{n} h_i r_i + c_{de} \sum_{i=1}^{n} d_i r_i\right)$

As described in Section 4.2, Hamiltonian pathfinding problems (including disassembly problems) can be transformed into the shortest pathfinding problem. The Q-compression method is executed on the original graph in the CSPP case and on a transformed graph in the other two cases. In CSHPP, the transformed graph vertices correspond to a feasible subpath that ends in a specific vertex in the original graph. In DLBP the last vertex of the sub-path has no influence. Therefore, the transformed graph's vertices correspond to the feasible subpaths only independently from the previous vertex.

The edges represent the connectivity of two vertices: whether a feasible transition exists from one vertex to the other. Edges integrate information about vertex connectivity in the CSPP and CSHPP cases and the precedence graph in the DLBP case.

The constraints describe the restrictions on range utilization in the CSPP and CSHPP cases and the workstation utilization in the DLBP case.

In the CSPP and CSHPP cases, the objective stands for a weighted sum of the total distances of the travelled path and the number of performed ranges. In the DLBP case, the objective is more complex: it is a weighted sum of the quadratic idle time of the workstations and the products of component disassembly order and their hazardousness and demand values.

Aligning with the traditional setup of RL solutions, I use the opposite of objectives for reward. Therefore, optimization problems become maximization tasks. I present my results by retransforming rewards into original objectives.

4.4.1 Constrained shortest path finding use case

The first use case is a simple shortest pathfinding problem. Figure 4.3 presents a directed graph. The edges are labelled with their average lengths. However, these

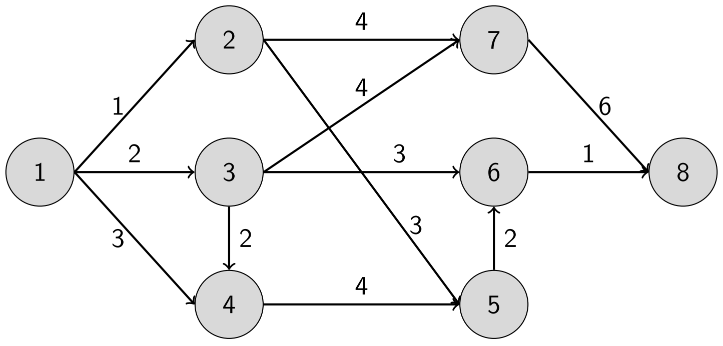


FIGURE 4.3: Distance graph of shortest pathfinding use case

distances are stochastic, which was simulated by multiplying random values from a log-normal distribution with parameter values of $\mu = 0$ and $\sigma = 0.01$. The goal is to find the shortest path from vertex 1 to vertex 8, considering an additional constraint: the traveller can only go through $L = 4.5$ units in one go. Man can easily see that the expected overall shortest path has a length of 7 units. The path $1 \to 3 \to 6 \to 8$ should be divided into three sub-paths to remain within the range limit, while the path $1 \to 2 \to 5 \to 6 \to 8$ should only be split into two.

The objective function is a slightly modified version of Equation 4.3 by standing for three terms:

$$\min z = \big(c_d \sum_{j=1}^{l} |\mathcal{P}_j| + c_o \sum_{j=1}^{l} (|\mathcal{P}_j > L) + c_r l\big) \tag{4.7}$$

The first term measures the total length of performed distances. The second term counts the ranges that exceed the limit value of L. Converting the range-limit constraint into an objective term transforms it into a soft condition. The third term describes the number of ranges into which the entire route was divided. The final objective function is a weighted sum of these three terms. In my use case, I set the c_d distance proportional cost to 1, the c_o overutilization cost to 100, and the c_r range cost to 10.

There were 20 simulations performed with 500 episodes each for both the Q-compression and grid-based methods. The granularity of the grid was set to 0.01. Figure 4.4 shows the distance distribution histogram by edges with a granularity of 0.01 unit.

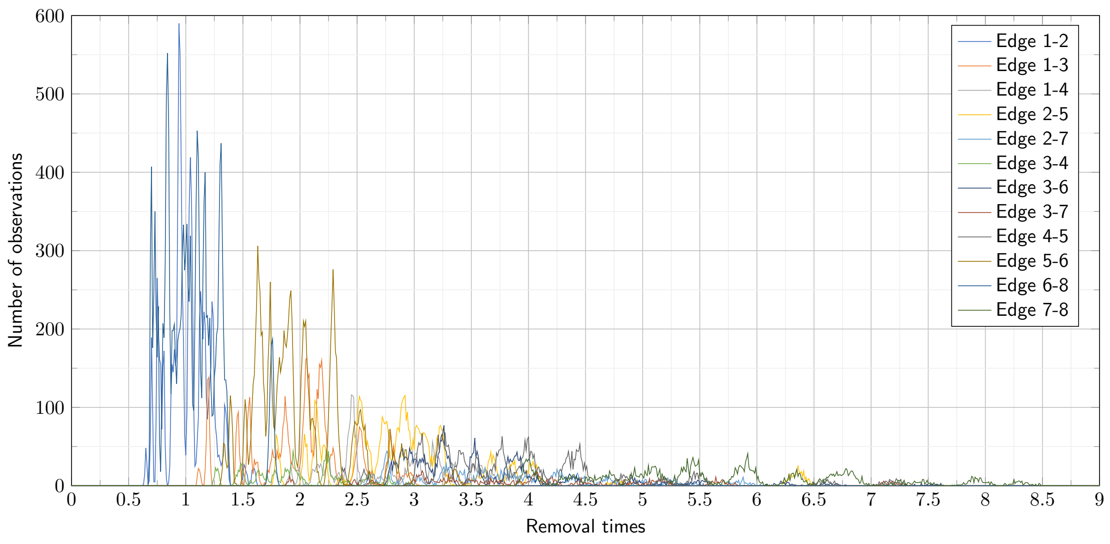


FIGURE 4.4: Removal time distributions by personal computer components

The ϵ-Greedy decision mechanism was applied with a linearly decreasing ϵ-strategy. The frequency of updating the discretization and the Q-table was set to 50 episodes. Figure 4.5 presents the simulation results.

The optimal solution to the deterministic problem has an objective value of 27. The readers can observe that in some cases, the empirical episode reward is lower than this, which comes from the stochastic property of the problem. Both the grid-based discretization method and the Q-compression method improve their performance as ϵ approaches 0. Still, the Q-compression finds a better objective value on average than the grid-based method (29.26 vs. 36.95). Table 4.2 summarizes the increase in Q-table size by episodes for the grid-based and Q-compression methods. It shows that the Q-table continuously grows using the

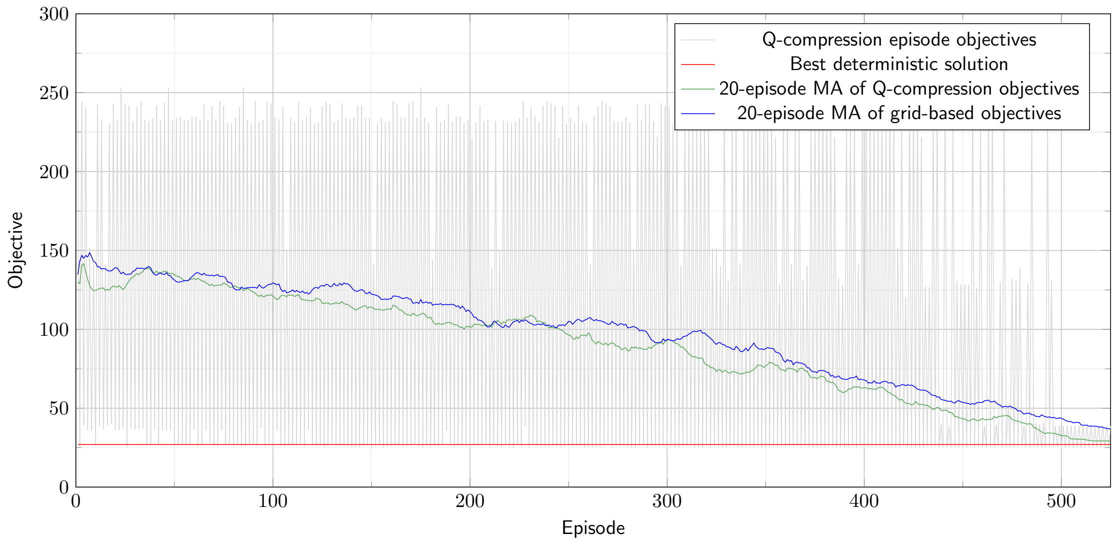


FIGURE 4.5: Q-compression method performance on constrained shortest pathfinding use case

grid-based discretization method, even with decreasing momentum. In contrast, the Q-compression method keeps the Q-table compact, which definitely supports the better learning capability observed in the achieved objective values.

TABLE 4.2: Q-table size comparison for CSPP by episodes

Discretization method	Episode									
	50	100	150	200	250	300	350	400	450	500
Grid-based	135	203	254	292	320	340	356	366	372	375
Q-compression	39	42	43	43	43	43	43	43	43	43

4.4.2 Constrained Hamiltonian path finding use case

The second use case is a Hamiltonian pathfinding problem [469]. Figure 4.6 presents a directed graph. Since no distances are declared from the edges, the average length of all edges was set to 1. To ensure the feasibility of the problem, non-existing edges are replaced with new edges with an average length of 100. Similarly to the CHPP use case, multiplicative stochastic noise was applied from a log-normal distribution with values of the parameters $\mu = 0$ and $\sigma = 0.05$. The goal is to find the continuous path from the vertex 1 to the vertex 5 by visiting all the other vertices exactly once. The range limit is set to $L = 10$. Note that an

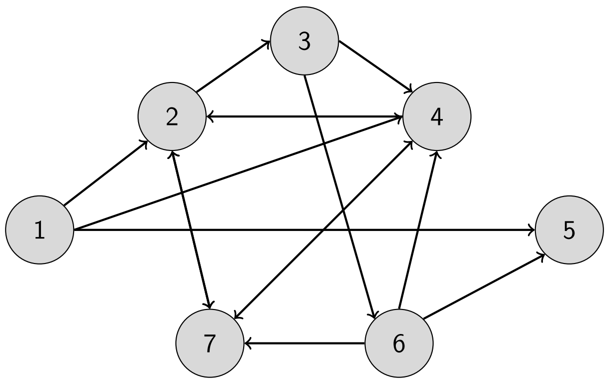


FIGURE 4.6: Directed graph of Hamiltonian pathfinding use case

entire feasible Hamiltonian path does not exceed the range limit. Furthermore, it is easy to validate that $1 \rightarrow 4 \rightarrow 7 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 5$ is a suitable solution (and the only one).

The objective function is identical to the previous one formulated in Equation 4.7. The weighting parameters are as follows: $c_d = 1$, $c_u = 100$, $c_r = 10$. The other settings are identical to the descriptions of the CSPP use case.

There were 20 simulations performed with 2,000 episodes each for both the Q-compression and grid-based methods. The granularity of the grid was set to 0.1. The frequency of updating the discretization and the Q-table was set to 100 episodes. Figure 4.7 presents the simulation results. The optimal solution to the

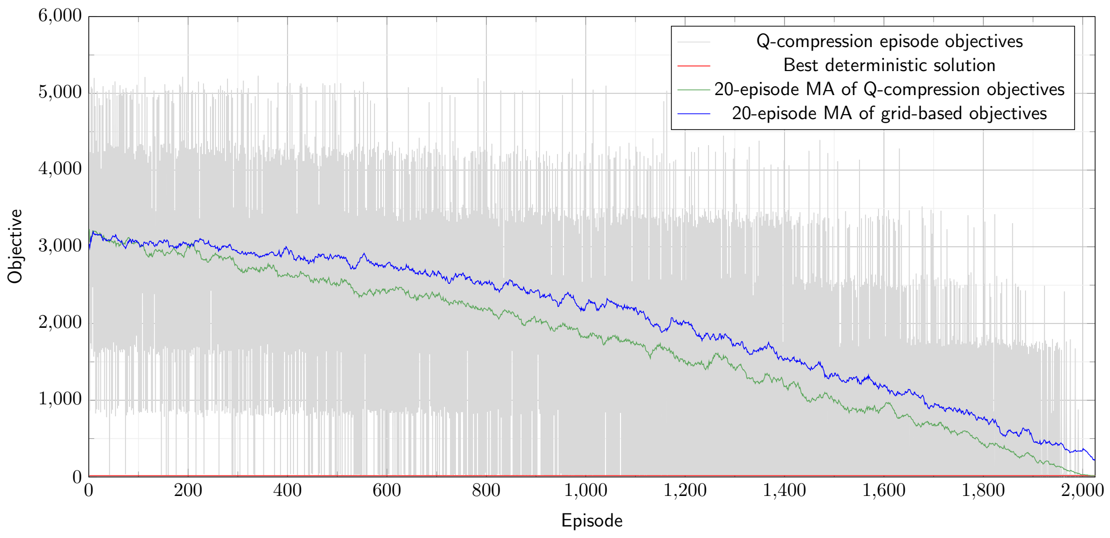


FIGURE 4.7: Q-compression method performance on constrained Hamiltonian pathfinding use case

deterministic problem has an objective value of 16. The Q-compression method approaches the optimal solution better by reaching an average objective value of 17.01 than the grid-based method, which produces an average objective value of 224.38. However, both show constant convergence to it. Table 4.3 shows the size of the Q-table by episodes. The readers can observe that the grid-based method grows the Q-table to an average size of $3,647$ records, whereas the Q-compression method leads to less than half the size having $1,735$ records on average.

TABLE 4.3: Q-table size comparison for CHPP by episodes

Discretization	Episode																			
method	100	200	300	400	500	600	700	800	900	1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000
Grid-based	649	1,127	1,526	1,863	2,167	2,425	2,650	2,844	3,008	3,145	3,266	3,362	3,442	3,504	3,554	3,589	3,613	3,630	3,640	3,647
Q-compression	411	615	788	944	1,093	1,223	1,339	1,428	1,504	1,563	1,615	1,651	1,678	1,698	1,713	1,723	1,730	1,733	1,735	1,735

4.4.3 Disassembly line balancing use case

The third use case is a computer disassembling problem from the literature [440] [449]. The problem was described in detail in Section 3.4.1. Similarly to the previous use cases, multiplicative stochastic noise was applied from a log-normal distribution with $\mu = 0$ and $\sigma = 0.05$ parameter values. There are eight salvageable components of a PC. Parts, removal times, demand values and hazardousness indicators are collected in Table 3.3.

A precedence graph describes the logical dependencies of the order of the disassembly task in Figure 3.4. The objective function comes from Equation 4.6 with a minor modification to sanction over-utilization of the workstation:

$$\min z = \big(c_i \textstyle\sum_{j=1}^{l}(t_c - |\tilde{\mathcal{H}}_j|)^2 + c_o l \min(\max(|\tilde{\mathcal{H}}_j| - t_c), 0) + c_h \sum_{j=1}^{n} \tilde{h}_j j + c_{de} \sum_{j=1}^{n} \tilde{d}_j j \qquad (4.8)$$

The reader should be reminded that the first term is quadratic idle time. It minimizes both the number of workstations used for disassembling the products and their imbalance. The second term is the soft criterion to keep the workstation utilization under the limit. If any workstations break the limit, a significant penalty will be imposed. The third and fourth terms force removing hazardous components and components with a higher demand value earlier. Following the referenced literature, the weights of c_i, c_h, and c_d are set at 1, while c_o has a value of 100. The cycle time (corresponding to the range limit L of previous use cases) is declared externally: $t_c = 40$.

I performed 20 simulations with 1,000 episodes each for both the Q-compression and grid-based methods. The granularity of the grid was set to 0.1. The frequency of updating the discretization and the Q-table was set to 100 episodes. Figure 4.8 shows the removal time distribution histogram with unit granularity 0.1 by the components of the personal computer.

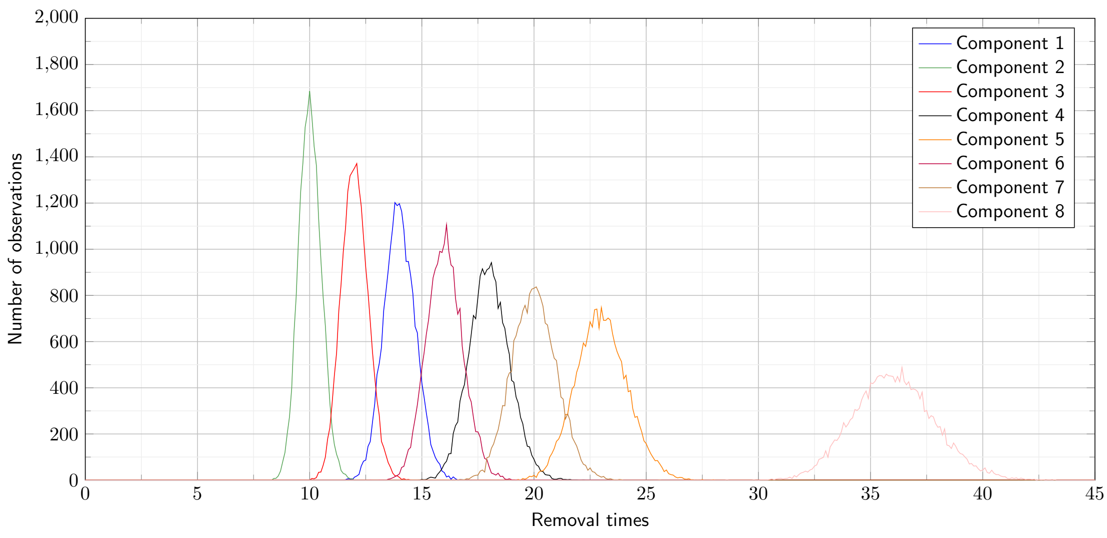


FIGURE 4.8: Removal time distributions by personal computer components

Next, Figure 4.9 presents the simulation results. The optimal solution to the deterministic problem has an objective value of 19, 065. In this use case, the reward curves of the two discretization methods diverge. The Q-compression method significantly better approaches the optimal solution by reaching an average objective value of 19, 122.29. In contrast, the grid-based method produces an average objective value of 22, 747.21.

As Table 4.4 shows, the grid-based method grows the Q-table to an average size of 2, 672 records, while the Q-compression method keeps the Q-table very compact by having 146 records on average, which definitely supports the better learning capability that can be observed in the achieved objective values.

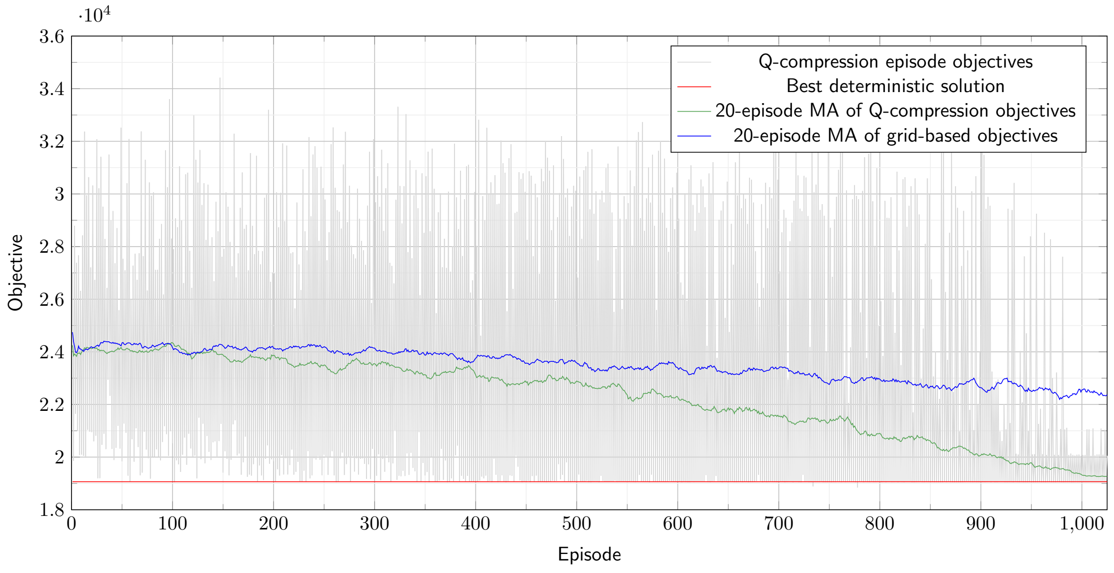


FIGURE 4.9: Q-compression method performance on personal computer disassembly use case

TABLE 4.4: Q-table size comparison for DLBP by episodes

Discretization method	Episode 100	200	300	400	500	600	700	800	900	1,000
Grid-based	702	1,158	1,522	1,834	2,087	2,320	2,485	2,609	2,716	2,807
Q-compression	144	151	152	150	147	147	146	146	146	146

4.5 Summary of constrained graph traversal problems

In this chapter, I gave an overview of some graph traversal problems and showed that they have a common root, namely the constrained shortest path finding problem. Then, a multistep Q-table compression method was presented to provide a human-interpretable solution for mixed-integer problems. The main result of this method is that it provides an algorithm to find an optimal abstraction level by providing a reduced state space with significantly smaller element sizes. I demonstrated the ability of my method on stochastic restricted graph traversal use cases and found that it produces a more compact Q-table than the grid-based discretization technique. As the optimization problem is more complex, the difference in performance is more significant.

I tested the methodology's performance on three selected use cases: a constrained stochastic shortest pathfinding problem, a constrained stochastic Hamiltonian pathfinding problem, and a stochastic disassembly line balancing problem. In all three use cases, I compared the performance of my developed method for discretizing the mixed continuous-discrete state space to the classical grid-based partitioning approach. My DBSCAN-based Q-table compression method achieves a better objective function value while keeping the state space size manageable.

Chapter 5

Conclusions and future research

5.1 Conclusions

In Chapter 2, I prepared a comprehensive review of the literature on RL methods and their uses in Industry 4.0 tasks. After that, I created a categorization for both RL methods and Industry 4.0 problems based on their nature and topics, and by showing the general trends that can be discovered from publications, I summarized which RL techniques promise more significant potential for which types of tasks. Then, based on the prerequisites and operating mechanisms of the different RL techniques, I prepared a general methodological guide in a decision tree structure to select the appropriate RL method based on the problem's properties.

Then, in Chapter 3, I pointed out that with the transition to a circular economy, the development and optimization of disassembly chains play an increasingly important role. After a literature review of the problem's various formalizations and solution methods, I created a Q-table-based reinforcement learning solution, which proved to be especially suitable for a deeper examination of the procedure's parameters and for exploring its practical limitations. During the research, it was possible to effectively combine the problem-reflective heuristics with the general Q-learning methods, which I showed that satisfying the RL formalization, the heuristic can show a general way to develop Q-table-based solutions for mixed-integer optimization tasks.

Finally, in Chapter 4, I focused on developing the Q-compression method and framing our research so far into a more general concept.

I analyzed a special class of graph traversal problems, where the distances are stochastic, and the agent is restricted to take a limited range in a time. I showed that constrained shortest Hamiltonian pathfinding problems and disassembly line balancing problems both belong to constrained shortest pathfinding problems, which can be represented as mixed-integer optimization problems.

Reinforcement learning methods have proven their efficiency in multiple complex problems. However, researchers concluded that the learning time increases radically by growing the state- and action spaces. In continuous cases, approximation techniques are used, but these methods cannot be applied in mixed-integer searching spaces.

I developed the Q-table compression technique as a multi-step method with dimension reduction, state fusion, and space compression techniques that project a mixed-integer optimization problem into a discrete one. Then, the RL agent is trained using an extended Q-learning method to provide a human-interpretable model for optimal action selection.

My approach was tested in selected constrained stochastic graph traversal use cases, and comparative results were collected.

In general, reinforcement learning methods are easily applicable to various industrial problems by providing easily implementable solutions for complex optimization tasks. Although the global optimum cannot be guaranteed, they deliver a continuously improved optimal solution from an early stage.

5.2 Thesis findings

The following list contains my new scientific results in three thesis findings.

1. **A multidimensional classification of reinforcement learning (RL) methods for applications in Industry 4.0 facilitates the identification of prevailing research trends. Furthermore, a questionnaire-based guideline supports the adoption of an appropriate methodology for the setup of RL solutions and helps to select the most suitable RL method to address specific problems.**

   1.1 I performed a systematic literature review that included keyword standardization and categorization into three major dimensions: principles captured, industrial field, and mathematical approaches to the application methodology.

   1.2 I made an analysis to identify the main research trends and the correlations between the problem types, the applied RL methods, and the industrial fields.

   1.3 Based on these, I created a questionnaire guideline to support the adoption of an appropriate methodology for the setup of RL solutions and help select the most suitable RL method to address specific problems.

   1.4 The guideline can be improved to being a fundamental basis of an automated RL method selection application.

   Related publications: [R1].

2. **The newly developed reinforcement learning-based dlOptRL algorithm can solve disassembly line balancing problems and effectively supports the optimal operation of sustainable disassembly lines.**

   2.1 I prepared the general formulation of the disassembly line balancing problem for the reinforcement learning methods including the state- and action spaces and the reward function.

   2.2 I developed the DLOPTRL algorithm that belongs to the Heuristically Accelerated Reinforcement Learning methods to solve disassembly line optimization problems more effectively than the original reinforcement learning method.

2.3 Hence there are limitations in most available linear and quadratic solvers, I reformulated the disassembly line balancing problem as a mixed-integer quadratic optimization problem using significantly fewer decision variables to provide a reference solution to evaluate the performance of the dlOptRL algorithm.

2.4 I derived a step-by-step procedure for generating the RL formulation directly from the MIQP formulation.

2.5 I demonstrated the applicability of the dlOptRL algorithm in small- and medium-scale disassembly line optimization use cases. I pointed out that my algorithm provides a real alternative compared to the classical Mixed-Integer Quadratic Programming solution.

Related publications: [R2], [R3].

3. **The Q-compression method is a newly developed, generally applicable reinforcement learning-based approach for solving mixed-integer optimization problems using a discrete dynamic representation of the state space across a wide range of complex optimization problems.**

3.1 I defined the class of restricted stochastic graph traversal problems and showed that many scheduling and routing problems belong to that. I also defined the exact method to transform the problems into a constrained stochastic shortest pathfinding problem and, hence, standardize the formulation.

3.2 I developed the Q-compression method to find a dynamic discrete representation of a mixed-integer optimization problem using the DBSCAN algorithm and solve it using the reinforcement learning method.

3.3 The Q-compression method improves the built-in heuristic of the dlOptRL algorithm to reduce the action space and provides a human-interpretable model without time-consuming preliminary steps.

3.4 Finally, I demonstrated the usability of the algorithm in different selected problems and verified its advance to the grid-based discretization method.

Related publications: [R4], [R5].

5.3 Utilization of new scientific results

The overview and questionnaire guidelines in my first thesis help researchers and developers shorten the preparation time by allowing them to focus on the most relevant methods. They can also be the basis for a later automatic method selection procedure.

My second thesis delivered the dlOptRL algorithm, which offers a complete solution for disassembly line balancing problems that can learn directly from an ongoing operation or a simulated virtual twin.

My third thesis declares the constrained stochastic graph traversal problem class, which involves several essential optimization problems and can definitely be extended by further ones. This class enables the easy transfer of a solution method from one member problem to another. The Q-compression method presents a human-interpretable solution without time-consuming preliminary steps, and therefore, its implementation time can be significantly reduced. The algorithm can potentially adapt to changing environments by modifying its observation time window.

5.4 Open questions and further research ideas

In the course of my research, I managed to identify several topics promising further significant potential for results. In this section, I present the open questions on which I would like to continue my research during the upcoming periods.

1. I presented a questionnaire guideline to help guide the follow-up of an appropriate methodology to set up an RL solution and to support selecting the most suitable RL method to address specific problems. This can be extended to a more general RL setup wizard tool by involving the state- and action-space setup, the reward function specification, and after the appropriate RL method selection, its hyperparameter setup.

2. An interesting and deep problem is to modify the dlOptRL algorithm for a multi-agent setup. If the different agents work on a common problem, the reward function should balance the agents' performance to find an overall

optimum. Otherwise, agents become selfish and are not allowed to perform worse individually but cannot reach a better overall performance.

3. In the other case, efficient knowledge transfer is far from trivial when the agents work separately but in a similar or identical environment. There are multiple choices: publishing the observation history enables the agents to learn faster, but if the environments have differences, then the agents should recognize this, and the external experience should be processed only conditionally.

4. The relevancy of the constrained stochastic graph traversal problem class could be increased if further problem types could be traced back to the shortest pathfinding problem.

5. The applicability of the Q-compression method could be increased by developing a solution to estimate how close the current optimum is to the global one. Of course, the global optimum cannot be determined because it is generally a computationally complex problem. If it were feasible, then the Q-compression method would be unnecessary. So, the question is, how could such an estimation be determined?

6. An obvious possibility would be to transform Q-compression into an adaptive learning solution. Its relevance comes from the challenge of changing environments. The first option is to use a moving time window of observation history in the Q-table update phase. In this case, the optimal width of the time window is an open question and should depend on the intensity of the environment changing, but relevant indicators should also be constructed to measure it.

I would be delighted if my supervisors and colleagues could support my further research on these open questions.

Acronyms

AC	Actor-critic
ADP	Asynchronous dynamic programming
AI	Artificial Intelligence
BC	Behavioural cloning
CCL	Cloud Control Layer
CHPP	Constrained Hamiltonian Pathfinding Problem
CPS	Cyber-Physical Systems
CSHPP	Constrained Shortest Hamiltonian Pathfinding Problem
CSPP	Constrained Shortest Pathfinding Problem
DDLP	Deep Deterministic Learning Policy
DLBP	Disassembly Line Balancing Problem
DoS	Denial-of-Service (attack type)
DP	Dynamic Programming
DQN	Deep Q-network
ER	Experience Replay
ES	Experience sharing
HARL	Heuristically Accelerated Reinforcement Learning
HCEQ	Hierarchical Correlated Q-learning
I4.0	Industry 4.0
IL	Imitation learning
IoT	Internet of things
IRL	Inverse Reinforcement Learning
IS	Importance sampling
LP	Linear Programming

MAAC	Multi-agent actor-critic
MABP	Multi-armed Bandit Problem
MARL	Multi-agent Reinforcement Learning
MC	Monte-Carlo (learning method)
MDP	Markov Decision Process
MG	Markov Games
MILP	Mixed-Integer Linear Programming
MIQP	Mixed-Integer Quadratic Programming
ML	Machine Learning
MMDP	Multi-agent Markov Decision Process
NE	Nash Equilibrium
PG	Policy gradient
POMG	Partially observable Markov game
PRISMA-P	Preferred Reporting Items for Systematic reviews and Meta-Analysis Protocols
PS	Prioritized sweeping
PV	Present Value
RL	Reinforcement Learning
SKU	Standardized Keyword Unit
TD	Temporal-Difference (learning method)
TERL	Transfer Expert Reinforcement Learning
TSP	Traveling Salesman Problem
UCL	User Control Layer
VF	Value function
VRP	Vehicle Routing Problem

Bibliography

Related publications to theses

[R1] Tamás Kegyes, Zoltán Süle, and János Abonyi. The applicability of reinforcement learning methods in the development of industry 4.0 applications. *Complexity*, 2021:1–31, 2021.

[R2] Tamás Kegyes, Zoltán Süle, and János Abonyi. Disassembly line optimization with reinforcement learning. *Central European Journal of Operations Research*, pages 1–28, 2024.

[R3] Tamás Kegyes, Zoltán Süle, and János Abonyi. Incorporation of heuristic search into q-learning. In *Short papers of VOCAL 2022 (9th VOCAL Optimization Conference: Advanced Algorithms)*, Budapest, Hungary, 2022.

[R4] Tamás Kegyes, Alex Kummer, Zoltán Süle, and János Abonyi. Generally applicable q-table compression method and its application for constrained stochastic graph traversal optimization problems. *Information*, 15(4):193, 2024.

[R5] Tamás Kegyes, Alex Kummer, Zoltán Süle, and János Abonyi. Improvements of the q-compression method for constrained stochastic graph traversal problems. In *10th VOCAL Optimization Conference: Advanced Algorithms: Conference Program)*, Budapest, Hungary, 2024.

Further publications

[F1] Tamás Kegyes, Zoltán Süle, and János Abonyi. Machine learning-based decision support framework for cbrn protection. *Heliyon*, 2024.

[F2] Kegyes Tamás, Süle Zoltán, and Abonyi János. Az információmenedzsment szerepe az abv-védelemben. *Nemzetbiztonsági Szemle*, 11(1):62–77, 2023.

[F3] Tamás Kegyes, Zoltán Süle, and János Abonyi. Szétszerelési láncok optimálisa megerősítéses tanulással. In *Magyar Operációkutatási Konferencia: Absztraktok könyve*, Budapest, Hungary, 2021.

References

[1] Y. Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

[2] V. Roblek, M. Meško, and A. Krapež. A complex view of industry 4.0. *SAGE Open*, 6(2), 2016.

[3] J. Posada, C. Toro, I. Barandiaran, D. Oyarzun, D. Stricker, R. De Amicis, E.B. Pinto, P. Eisert, J. Döllner, and Jr. Vallarino, I. Visual computing as a key enabling technology for industrie 4.0 and industrial internet. *IEEE Computer Graphics and Applications*, 35(2):26–40, 2015.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018.

[5] R. Csalódi, Z. Süle, S. Jaskó, T. Holczinger, and J. Abonyi. Industry 4.0-driven development of optimization algorithms: A systematic overview. *Complexity*, 2021, 2021.

[6] K.-D. Thoben, S.A. Wiesner, and T. Wuest. "industrie 4.0" and smart manufacturing-a review of research issues and application examples. *International Journal of Automation Technology*, 11(1):4–16, 2017.

[7] J. Mattioli, P. Perico, and P.-O. Robic. In *Improve Total Production Maintenance with Artificial Intelligence*, pages 56–59. Institute of Electrical and Electronics Engineers Inc., 2020.

[8] A. Dolgui, D. Ivanov, S.P. Sethi, and B. Sokolov. Scheduling in production, supply chain and industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. *International Journal of Production Research*, 57(2):411–432, 2019.

[9] A. Diez-Olivan, J. Del Ser, D. Galar, and B. Sierra. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0. *Information Fusion*, 50:92–111, 2019.

[10] Radhey Shyam and Riya Chakraborty. Machine learning and its dominant paradigms. *Journal of Advancements in Robotics*, 8(2):1–10, 2021.

[11] Martin L Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):1127–1137, 1978.

[12] F. Castaño, G. Beruvides, A. Villalonga, and R.E. Haber. Self-tuning method for increased obstacle detection reliability based on internet of things lidar sensor models. *Sensors (Switzerland)*, 18(5), 2018.

[13] A. Ferdowsi and W. Saad. Deep learning for signal authentication and security in massive internet-of-things systems. *IEEE Transactions on Communications*, 67(2):1371–1387, 2019.

[14] M. Chu, H. Li, X. Liao, and S. Cui. Reinforcement learning-based multiaccess control and battery prediction with energy harvesting in iot systems. *IEEE Internet of Things Journal*, 6(2):2009–2020, 2019.

[15] D. Nallaperuma, R. Nawaratne, T. Bandaragoda, A. Adikari, S. Nguyen, T. Kempitiya, D. De Silva, D. Alahakoon, and D. Pothuhera. Online incremental machine learning platform for big data-driven smart traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4679–4690, 2019.

[16] S. Shresthamali, M. Kondo, and H. Nakamura. Adaptive power management in solar energy harvesting sensor node using reinforcement learning. *ACM Transactions on Embedded Computing Systems*, 16(5s), 2017.

[17] Y.P. Pane, S.P. Nageshrao, J. Kober, and R. Babuška. Reinforcement learning based compensation methods for robot manipulators. *Engineering Applications of Artificial Intelligence*, 78:236–247, 2019.

[18] X. Wang, C. Wang, X. Li, V.C.M. Leung, and T. Taleb. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*, 7(10):9441–9455, 2020.

[19] L. Li, Y. Xu, J. Yin, W. Liang, X. Li, W. Chen, and Z. Han. Deep reinforcement learning approaches for content caching in cache-enabled d2d networks. *IEEE Internet of Things Journal*, 7(1):544–557, 2020.

[20] W. Hu, Y. Wen, K. Guan, G. Jin, and K.J. Tseng. Itcm: Toward learning-based thermal comfort modeling via pervasive sensing for smart buildings. *IEEE Internet of Things Journal*, 5(5):4164–4177, 2018.

[21] B. Chen, J. Wan, Y. Lan, M. Imran, D. Li, and N. Guizani. Improving cognitive ability of edge intelligent iiot through machine learning. *IEEE Network*, 33(5):61–67, 2019.

[22] C. Zhang, C. Gupta, A. Farahat, K. Ristovski, and D. Ghosh. Equipment health indicator learning using deep reinforcement learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11053 LNAI:488–504, 2019.

[23] A. Kawewong, Y. Honda, M. Tsuboyama, and O. Hasegawa. Reasoning on the self-organizing incremental associative memory for online robot path planning. *IEICE Transactions on Information and Systems*, E93-D(3):569–582, 2010.

[24] W. Xiong, Z. Lu, B. Li, Z. Wu, B. Hang, J. Wu, and X. Xuan. A self-adaptive approach to service deployment under mobile edge computing for autonomous driving. *Engineering Applications of Artificial Intelligence*, 81:397–407, 2019.

[25] M. Chu, X. Liao, H. Li, and S. Cui. Power control in energy harvesting multiple access system with reinforcement learning. *IEEE Internet of Things Journal*, 6(5):9175–9186, 2019.

[26] A. Ismail and V. Cardellini. Decentralized planning for self-adaptation in multi-cloud environment. *Communications in Computer and Information Science*, 508:76–90, 2015.

[27] B. Wang, Y. Sun, T.Q. Duong, L.D. Nguyen, and L. Hanzo. Risk-aware identification of highly suspected covid-19 cases in social iot: A joint graph

theory and reinforcement learning approach. *IEEE Access*, 8:115655–115661, 2020.

[28] F. Zhou, Q. Yang, K. Zhang, G. Trajcevski, T. Zhong, and A. Khokhar. Reinforced spatiotemporal attentive graph neural networks for traffic forecasting. *IEEE Internet of Things Journal*, 7(7):6414–6428, 2020.

[29] L. Roveda, J. Maskani, P. Franceschi, A. Abdi, F. Braghin, L. Molinari Tosatti, and N. Pedrocchi. Model-based reinforcement learning variable impedance control for human-robot collaboration. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 100(2):417–433, 2020.

[30] H. Wu, Z. Zhang, C. Jiao, C. Li, and T.Q.S. Quek. Learn to sense: A meta-learning-based sensing and fusion framework for wireless sensor networks. *IEEE Internet of Things Journal*, 6(5):8215–8227, 2019.

[31] G. Vallathan, A. John, C. Thirumalai, S.K. Mohan, G. Srivastava, and J.C.-W. Lin. Suspicious activity detection using deep learning in secure assisted living iot environments. *Journal of Supercomputing*, 77(4):3242–3260, 2021.

[32] S. Chesney, K. Roy, and S. Khorsandroo. Machine learning algorithms for preventing iot cybersecurity attacks. *Advances in Intelligent Systems and Computing*, 1252 AISC:679–686, 2021.

[33] K.H.K. Reddy, A.K. Luhach, B. Pradhan, J.K. Dash, and D.S. Roy. A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. *Sustainable Cities and Society*, 63, 2020.

[34] M.S. Munir, S.F. Abedin, N.H. Tran, Z. Han, E. Huh, and C.S. Hong. Risk-aware energy scheduling for edge computing with microgrid: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Network and Service Management*, 18(3):3476–3497, 2021.

[35] B. Wang, Y. Sun, M. Sun, and X. Xu. Game-theoretic actor-critic-based intrusion response scheme (gtac-irs) for wireless sdn-based iot networks. *IEEE Internet of Things Journal*, 8(3):1830–1845, 2021.

[36] E. Baccour, A. Erbad, A. Mohamed, F. Haouari, M. Guizani, and M. Hamdi. Rl-opra: Reinforcement learning for online and proactive resource allocation of crowdsourced live videos. *Future Generation Computer Systems*, 112:982–995, 2020.

[37] S.-G. Choi and S.-B. Cho. Bayesian networks + reinforcement learning: Controlling group emotion from sensory stimuli. *Neurocomputing*, 391:355–364, 2020.

[38] K. Lepenioti, M. Pertselakis, A. Bousdekis, A. Louca, F. Lampathaki, D. Apostolou, G. Mentzas, and S. Anastasiou. Machine learning for predictive and prescriptive analytics of operational data in smart manufacturing. *Lecture Notes in Business Information Processing*, 382 LNBIP:5–16, 2020.

[39] L. Lei, Y. Tan, G. Dahlenburg, W. Xiang, and K. Zheng. Dynamic energy dispatch based on deep reinforcement learning in iot-driven smart isolated microgrids. *IEEE Internet of Things Journal*, 8(10):7938–7953, 2021.

[40] Z.-Y. Wu, M. Ismail, E. Serpedin, and J. Wang. Data-driven link assignment with qos guarantee in mobile rf-optical hetnet of things. *IEEE Internet of Things Journal*, 7(6):5088–5102, 2020.

[41] S. Sun, X. Li, M. Liu, B. Yang, and X. Guo. Dnn inference acceleration via heterogeneous iot devices collaboration. *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, 57(4):709–722, 2020.

[42] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu. Iot security techniques based on machine learning: How do iot devices use ai to enhance security? *IEEE Signal Processing Magazine*, 35(5):41–49, 2018.

[43] W. Liang, W. Huang, J. Long, K. Zhang, K.-C. Li, and D. Zhang. Deep reinforcement learning for resource protection and real-time detection in iot environment. *IEEE Internet of Things Journal*, 7(7):6392–6401, 2020.

[44] X. Zhou, W. Liang, K.I.-K. Wang, H. Wang, L.T. Yang, and Q. Jin. Deep-learning-enhanced human activity recognition for internet of healthcare things. *IEEE Internet of Things Journal*, 7(7):6429–6438, 2020.

[45] F. Castaño, S. Strzełczak, A. Villalonga, R.E. Haber, and J. Kossakowska. Sensor reliability in cyber-physical systems using internet-of-things data: A review and case study. *Remote Sensing*, 11(19), 2019.

[46] S. Tu, M. Waqas, S.U. Rehman, M. Aamir, O.U. Rehman, Z. Jianbiao, and C.-C. Chang. Security in fog computing: A novel technique to tackle an impersonation attack. *IEEE Access*, 6:74993–75001, 2018.

[47] B. Chatterjee, S. Sen, N. Cao, and A. Raychowdhury. Context-aware intelligence in resource-constrained iot nodes: Opportunities and challenges. *IEEE Design and Test*, 36(2):7–40, 2019.

[48] N. Aihara, K. Adachi, O. Takyu, M. Ohta, and T. Fujii. Q-learning aided resource allocation and environment recognition in lorawan with csma/ca. *IEEE Access*, 7:152126–152137, 2019.

[49] W. Seok and C. Park. Recognition of human motion with deep reinforcement learning. *IEIE Transactions on Smart Processing and Computing*, 7(3):245–250, 2018.

[50] Q. Hu, S. Lv, Z. Shi, L. Sun, and L. Xiao. Defense against advanced persistent threats with expert system for internet of things. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10251 LNCS:326–337, 2017.

[51] A. Gaddam, T. Wilkin, M. Angelova, and J. Gaddam. Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions. *Electronics (Switzerland)*, 9(3), 2020.

[52] R.S. Alonso, I. Sittón-Candanedo, R. Casado-Vara, J. Prieto, and J.M. Corchado. Deep reinforcement learning for the management of software-defined

networks and network function virtualization in an edge-iot architecture. *Sustainability (Switzerland)*, 12(14), 2020.

[53] Y. Meng, S. Tu, J. Yu, and F. Huang. Intelligent attack defense scheme based on dql algorithm in mobile fog computing. *Journal of Visual Communication and Image Representation*, 65, 2019.

[54] X. Ma and W. Shi. Aesmote: Adversarial reinforcement learning with smote for anomaly detection. *IEEE Transactions on Network Science and Engineering*, 8(2):943–956, 2021.

[55] Y. Liu, K.-F. Tong, and K.-K. Wong. Reinforcement learning based routing for energy sensitive wireless mesh iot networks. *Electronics Letters*, 55(17):966–968, 2019.

[56] Y. Huang, X. Guan, H. Chen, Y. Liang, S. Yuan, and T. Ohtsuki. Risk assessment of private information inference for motion sensor embedded iot devices. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(3):265–275, 2020.

[57] P. Saha and S. Mukhopadhyay. Multispectral information fusion with reinforcement learning for object tracking in iot edge devices. *IEEE Sensors Journal*, 20(8):4333–4344, 2020.

[58] P. Sun, Y. Dong, S. Yuan, and C. Wang. Preventive control policy construction in active distribution network of cyber-physical system with reinforcement learning. *Applied Sciences (Switzerland)*, 11(1):1–20, 2020.

[59] Q.-D. Ngo, H.-T. Nguyen, H.-L. Pham, H.H.-N. Ngo, D.-H. Nguyen, C.-M. Dinh, and X.-H. Vu. A graph-based approach for iot botnet detection using reinforcement learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12496 LNAI:465–478, 2020.

[60] S. Sree Dharinya and E.P. Ephzibah. Machine intelligence and automation: Deep learning concepts aiding industrial applications. *EAI/Springer Innovations in Communication and Computing*, pages 237–248, 2020.

[61] Y.H. Lai, Y.C. Chang, C.W. Tsai, C.H. Lin, and M.Y. Chen. Data fusion analysis for attention-deficit hyperactivity disorder emotion recognition with thermal image and internet of things devices. *Software - Practice and Experience*, 51(3):595–606, 2021.

[62] Z. Zhang, C. Li, S.L. Peng, and X. Pei. A new task offloading algorithm in edge computing. *Eurasip Journal on Wireless Communications and Networking*, 2021(1), 2021.

[63] L. Espinosa-Leal, A. Chapman, and M. Westerlund. Autonomous industrial management via reinforcement learning. *Journal of Intelligent and Fuzzy Systems*, 39(6):8427–8439, 2020.

[64] S. Khan, M. Farnsworth, R. McWilliam, and J. Erkoyuncu. On the requirements of digital twin-driven autonomous maintenance. *Annual Reviews in Control*, 50:13–28, 2020.

[65] S.A. Alghamdi. An effective strategy for fingerprint recognition based on pram's neural ature with data input mappings. *Advances in Intelligent Systems and Computing*, 439:623–634, 2016.

[66] H. Benaddi, K. Ibrahimi, A. Benslimane, and J. Qadir. A deep reinforcement learning based intrusion detection system (drl-ids) for securing wireless sensor networks and internet of things. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 317 LNICST:73–87, 2020.

[67] Y. Qin, Q. Xia, Z. Xu, P. Zhou, A. Galis, O.F. Rana, J. Ren, and G. Wu. Enabling multicast slices in edge networks. *IEEE Internet of Things Journal*, 7(9):8485–8501, 2020.

[68] R. Heartfield, G. Loukas, A. Bezemskij, and E. Panaousis. Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 16:1720–1735, 2021.

[69] S. Guo, Y. Qi, Y. Jin, W. Li, X. Qiu, and L. Meng. Endogenous trusted drl-based service function chain orchestration for iot. *IEEE Transactions on Computers*, 70, 2021.

[70] G. Wu. Uav-based interference source localization: A multimodal q-learning approach. *IEEE Access*, 7:137982–137991, 2019.

[71] R.K. Dhanaraj, K. Rajkumar, and U. Hariharan. Enterprise iot modeling: Supervised, unsupervised, and reinforcement learning. *EAI/Springer Innovations in Communication and Computing*, pages 55–79, 2020.

[72] D. Yu, P. Li, Y. Chen, Y. Ma, and J. Chen. A time-efficient multi-protocol probe scheme for fine-grain iot device identification. *Sensors (Switzerland)*, 20(7), 2020.

[73] L. Chen, Y. Xu, Z. Lu, J. Wu, K. Gai, P.C.K. Hung, and M. Qiu. Iot microservice deployment in edge-cloud hybrid environment using reinforcement learning. *IEEE Internet of Things Journal*, 8(16):12610–12622, 2021.

[74] K.-H. Phung, B. Lemmens, M. Goossens, A. Nowe, L. Tran, and K. Steenhaut. Schedule-based multi-channel communication in wireless sensor networks: A complete design and performance evaluation. *Ad Hoc Networks*, 26:88–102, 2015.

[75] T. Akazaki, S. Liu, Y. Yamagata, Y. Duan, and J. Hao. Falsification of cyber-physical systems using deep reinforcement learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10951 LNCS:456–465, 2018.

[76] K. Gai, M. Qiu, M. Liu, and H. Zhao. Smart resource allocation using reinforcement learning in content-centric cyber-physical systems. *Lecture Notes in*

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10699 LNCS:39–52, 2018.

[77] F. Jazayeri, A. Shahidinejad, and M. Ghobaei-Arani. Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach. *Journal of Ambient Intelligence and Humanized Computing*, 12(8):8265–8284, 2021.

[78] A. Pauna, I. Bica, F. Pop, and A. Castiglione. On the rewards of self-adaptive iot honeypots. *Annales des Telecommunications/Annals of Telecommunications*, 74(7-8):501–515, 2019.

[79] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho. Multiagent ddpg-based deep learning for smart ocean federated learning iot networks. *IEEE Internet of Things Journal*, 7(10):9895–9903, 2020.

[80] C. Shu, Z. Zhao, G. Min, J. Hu, and J. Zhang. Deploying network functions for multiaccess edge-iot with deep reinforcement learning. *IEEE Internet of Things Journal*, 7(10):9507–9516, 2020.

[81] V. Antuori, E. Hebrard, M.-J. Huguet, S. Essodaigui, and A. Nguyen. Leveraging reinforcement learning, constraint programming and local search: A case study in car manufacturing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12333 LNCS:657–672, 2020.

[82] R. Wu, J. Gong, W. Tong, and B. Fan. Network attack path selection and evaluation based on q-learning. *Applied Sciences (Switzerland)*, 11(1):1–13, 2021.

[83] S. Redhu and R.M. Hegde. Cooperative network model for joint mobile sink scheduling and dynamic buffer management using q-learning. *IEEE Transactions on Network and Service Management*, 17(3):1853–1864, 2020.

[84] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo. Thirty years of machine learning: The road to pareto-optimal wireless networks. *IEEE Communications Surveys and Tutorials*, 22(3):1472–1514, 2020.

[85] S. Vimal, M. Khari, R.G. Crespo, L. Kalaivani, N. Dey, and M. Kaliappan. Energy enhancement using multiobjective ant colony optimization with double q learning algorithm for iot based cognitive radio networks. *Computer Communications*, 154:481–490, 2020.

[86] X. Liu, J. Yu, J. Wang, and Y. Gao. Resource allocation with edge computing in iot networks via machine learning. *IEEE Internet of Things Journal*, 7(4):3415–3426, 2020.

[87] S.K. Sathya Lakshmi Preetha, R. Dhanalakshmi, and R. Kumar. An energy efficient framework for densely distributed wsns iot devices based on tree based robust cluster head. *Wireless Personal Communications*, 103(4):3163–3180, 2018.

[88] J. Liu, D. Li, and Y. Xu. Collaborative online edge caching with bayesian clustering in wireless networks. *IEEE Internet of Things Journal*, 7(2):1548–1560, 2020.

[89] M.K. Pandit, R.N. Mir, and M.A. Chishti. Adaptive task scheduling in iot using reinforcement learning. *International Journal of Intelligent Computing and Cybernetics*, 13(3):261–282, 2020.

[90] H. Li, K. Ota, and M. Dong. Deep reinforcement scheduling for mobile crowdsensing in fog computing. *ACM Transactions on Internet Technology*, 19(2), 2019.

[91] S.A. Khowaja and P. Khuwaja. Q-learning and lstm based deep active learning strategy for malware defense in industrial iot applications. *Multimedia Tools and Applications*, 2021.

[92] A. Sharif, J.P. Li, M.A. Saleem, G. Manogran, S. Kadry, A. Basit, and M.A. Khan. A dynamic clustering technique based on deep reinforcement learning

for internet of vehicles. *Journal of Intelligent Manufacturing*, 32(3):757–768, 2021.

[93] F. Hussain, R. Hussain, A. Anpalagan, and A. Benslimane. A new block-based reinforcement learning approach for distributed resource allocation in clustered iot networks. *IEEE Transactions on Vehicular Technology*, 69(3):2891–2904, 2020.

[94] K. Nivitha, A. Solaiappan, and P. Pabitha. Robust service selection through intelligent clustering in an uncertain environment. *Advances in Intelligent Systems and Computing*, 1167:325–332, 2021.

[95] T. Zhou, D. Tang, H. Zhu, and L. Wang. Reinforcement learning with composite rewards for production scheduling in a smart factory. *IEEE Access*, 9:752–766, 2021.

[96] J.H. Cho and H. Lee. Dynamic topology model of q-learning leach using disposable sensors in autonomous things environment. *Applied Sciences (Switzerland)*, 10(24):1–19, 2020.

[97] H. Qi, X. Mu, and Y. Shi. A task unloading strategy of iot devices using deep reinforcement learning based on mobile cloud computing environment. *Wireless Networks*, 26, 2020.

[98] X. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen. Space/aerial-assisted computing offloading for iot applications: A learning-based approach. *IEEE Journal on Selected Areas in Communications*, 37(5):1117–1129, 2019.

[99] Y. Wei, F.R. Yu, M. Song, and Z. Han. Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor-critic deep reinforcement learning. *IEEE Internet of Things Journal*, 6(2):2061–2073, 2019.

[100] F. Bu and X. Wang. A smart agriculture iot system based on deep reinforcement learning. *Future Generation Computer Systems*, 99:500–507, 2019.

[101] C.-C. Lin, D.-J. Deng, Y.-L. Chih, and H.-T. Chiu. Smart manufacturing scheduling with edge computing using multiclass deep q network. *IEEE Transactions on Industrial Informatics*, 15(7):4276–4284, 2019.

[102] H. Yang, A. Alphones, W.-D. Zhong, C. Chen, and X. Xie. Learning-based energy-efficient resource management by heterogeneous rf/vlc for ultra-reliable low-latency industrial iot networks. *IEEE Transactions on Industrial Informatics*, 16(8):5565–5576, 2020.

[103] T. Yu, B. Zhou, K.W. Chan, and E. Lu. Stochastic optimal cps relaxed control methodology for interconnected power systems using q-learning method. *Journal of Energy Engineering*, 137(3):116–129, 2011.

[104] R. Bonnefoi, L. Besson, C. Moy, E. Kaufmann, and J. Palicot. Multi-armed bandit learning in iot networks: Learning helps even in non-stationary settings. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 228:173–185, 2018.

[105] S. Shen, Y. Han, X. Wang, and Y. Wang. Computation offloading with multiple agents in edge-computing–supported iot. *ACM Transactions on Sensor Networks*, 16(1), 2019.

[106] A. Nassar and Y. Yilmaz. Reinforcement learning for adaptive resource allocation in fog ran for iot with heterogeneous latency requirements. *IEEE Access*, 7:128014–128025, 2019.

[107] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Communications Surveys and Tutorials*, 22(3):1722–1760, 2020.

[108] R. Ali, Y.A. Qadri, Y. Bin Zikria, T. Umer, B.-S. Kim, and S.W. Kim. Q-learning-enabled channel access in next-generation dense wireless networks for iot-based ehealth systems. *Eurasip Journal on Wireless Communications and Networking*, 2019(1), 2019.

[109] J. Zhang and J. Sun. A game theoretic approach to multi-channel transmission scheduling for multiple linear systems under dos attacks. *Systems and Control Letters*, 133, 2019.

[110] M. Kwon, J. Lee, and H. Park. Intelligent iot connectivity: Deep reinforcement learning approach. *IEEE Sensors Journal*, 20(5):2782–2791, 2020.

[111] M. Camelo, M. Claeys, and S. Latre. Parallel reinforcement learning with minimal communication overhead for iot environments. *IEEE Internet of Things Journal*, 7(2):1387–1400, 2020.

[112] P. Farhat, H. Sami, and A. Mourad. Reinforcement r-learning model for time scheduling of on-demand fog placement. *Journal of Supercomputing*, 76(1):388–410, 2020.

[113] X. Wei, J. Zhao, L. Zhou, and Y. Qian. Broad reinforcement learning for supporting fast autonomous iot. *IEEE Internet of Things Journal*, 7(8):7010–7020, 2020.

[114] S.-G. Choi and S.-B. Cho. Sensor information fusion by integrated ai to control public emotion in a cyber-physical environment. *Sensors (Switzerland)*, 18(11), 2018.

[115] G. Rjoub, J. Bentahar, O. Abdel Wahab, and A. Saleh Bataineh. Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency Computation*, 33(23), 2020.

[116] P. Loreti, L. Bracciale, and G. Bianchi. Stablesens: Sampling time decision algorithm for iot energy harvesting devices. *IEEE Internet of Things Journal*, 6(6):9908–9918, 2019.

[117] S. Shukla, M.F. Hassan, L.T. Jung, and A. Awang. Architecture for latency reduction in healthcare internet-of-things using reinforcement learning and fuzzy based fog computing. *Advances in Intelligent Systems and Computing*, 843:372–383, 2019.

[118] H. Rashtian and S. Gopalakrishnan. Balancing message criticality and timeliness in iot networks. *IEEE Access*, 7:145738–145745, 2019.

[119] Y. Dai, G. Wang, K. Muhammad, and S. Liu. A closed-loop healthcare processing approach based on deep reinforcement learning. *Multimedia Tools and Applications*, 79, 2020.

[120] C. Lork, W.-T. Li, Y. Qin, Y. Zhou, C. Yuen, W. Tushar, and T.K. Saha. An uncertainty-aware deep reinforcement learning framework for residential air conditioning energy management. *Applied Energy*, 276, 2020.

[121] M. Faraji Mehmandar, S. Jabbehdari, and H. Haj Seyyed Javadi. A dynamic fog service provisioning approach for iot applications. *International Journal of Communication Systems*, 33(14), 2020.

[122] J. Zhang, J. Du, Y. Shen, and J. Wang. Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach. *IEEE Internet of Things Journal*, 7(10):9303–9317, 2020.

[123] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang. Collaborative edge computing and caching with deep reinforcement learning decision agents. *IEEE Access*, 8:120604–120612, 2020.

[124] S. Chen, J. Wang, H. Li, Z. Wang, F. Liu, and S. Li. Top-down human-cyber-physical data fusion based on reinforcement learning. *IEEE Access*, 8:134233–134245, 2020.

[125] J. Leng, G. Ruan, Y. Song, Q. Liu, Y. Fu, K. Ding, and X. Chen. A loosely-coupled deep reinforcement learning approach for order acceptance decision of mass-individualized printed circuit board manufacturing in industry 4.0. *Journal of Cleaner Production*, 280, 2021.

[126] M. Mobasheri, Y. Kim, and W. Kim. Fog fragment cooperation on bandwidth management based on reinforcement learning. *Sensors (Switzerland)*, 20(23):1–15, 2020.

[127] G. Rjoub, O. Abdel Wahab, J. Bentahar, and A. Bataineh. A trust and energy-aware double deep reinforcement learning scheduling strategy for federated learning on iot devices. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12571 LNCS:319–333, 2020.

[128] J. Na, H. Zhang, X. Deng, B. Zhang, and Z. Ye. Accelerate personalized iot service provision by cloud-aided edge reinforcement learning: A case study on smart lighting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12571 LNCS:69–84, 2020.

[129] M. Tiwari, S. Misra, P.K. Bishoyi, and L.T. Yang. Devote: Criticality-aware federated service provisioning in fog-based iot environments. *IEEE Internet of Things Journal*, 8(13):10631–10638, 2021.

[130] A. Haldorai, A. Ramu, and M. Suriya. Organization internet of things (iots): Supervised, unsupervised, and reinforcement learning. *EAI/Springer Innovations in Communication and Computing*, pages 27–53, 2020.

[131] A. Musaddiq, R. Ali, J.-G. Choi, B.-S. Kim, and S.-W. Kim. Collision observation-based optimization of low-power and lossy iot network using reinforcement learning. *Computers, Materials and Continua*, 67(1):799–814, 2021.

[132] X. Zhou, X. Dong, Z. Laiping, K. Li, and T. Qiu. Learning-driven cloud resource provision policy for content providers with competitors. *IEEE Transactions on Cloud Computing*, 8, 2020.

[133] Y. Hao, M. Chen, H. Gharavi, Y. Zhang, and K. Hwang. Deep reinforcement learning for edge service placement in softwarized industrial cyber-physical system. *IEEE Transactions on Industrial Informatics*, 17(8):5552–5561, 2021.

[134] T. Park, N. Abuzainab, and W. Saad. Learning how to communicate in the internet of things: Finite resources and heterogeneity. *IEEE Access*, 4:7063–7073, 2016.

[135] M.G.R. Alam, M.M. Hassan, M.Z. Uddin, A. Almogren, and G. Fortino. Autonomic computation offloading in mobile edge for iot applications. *Future Generation Computer Systems*, 90:149–157, 2019.

[136] C.H. Liu, Q. Lin, and S. Wen. Blockchain-enabled data collection and sharing for industrial iot with deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 15(6):3516–3526, 2019.

[137] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo. Green resource allocation based on deep reinforcement learning in content-centric iot. *IEEE Transactions on Emerging Topics in Computing*, 8(3):781–796, 2020.

[138] L. Huang, S. Bi, and Y.-J.A. Zhang. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Transactions on Mobile Computing*, 19(11):2581–2593, 2020.

[139] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu. Iraf: A deep reinforcement learning approach for collaborative mobile edge computing iot networks. *IEEE Internet of Things Journal*, 6(4):7011–7024, 2019.

[140] S. Deng, Z. Xiang, P. Zhao, J. Taheri, H. Gao, J. Yin, and A.Y. Zomaya. Dynamical resource allocation in edge for trustable internet-of-things systems: A reinforcement learning method. *IEEE Transactions on Industrial Informatics*, 16(9):6103–6113, 2020.

[141] K. Gai and M. Qiu. Optimal resource allocation using reinforcement learning for iot content-centric services. *Applied Soft Computing Journal*, 70:12–21, 2018.

[142] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang. Federated learning-based computation offloading optimization in edge computing-supported internet of things. *IEEE Access*, 7:69194–69201, 2019.

[143] Y. Li, H. Ji, X. Li, and V.C.M. Leung. Dynamic channel selection with reinforcement learning for cognitive wlan over fiber. *International Journal of Communication Systems*, 25(8):1077–1090, 2012.

[144] S. Vimal, M. Khari, N. Dey, R.G. Crespo, and Y. Harold Robinson. Enhanced resource allocation in mobile edge computing using reinforcement learning based moaco algorithm for iiot. *Computer Communications*, 151:355–364, 2020.

[145] A. Alsarhan, A. Itradat, A.Y. Al-Dubai, A.Y. Zomaya, and G. Min. Adaptive resource allocation and provisioning in multi-service cloud environments. *IEEE Transactions on Parallel and Distributed Systems*, 29(1):31–42, 2018.

[146] Z. Wei, B. Zhao, J. Su, and X. Lu. Dynamic edge computation offloading for internet of things with energy harvesting: A learning method. *IEEE Internet of Things Journal*, 6(3):4436–4447, 2019.

[147] J. Wang, C. Jiang, K. Zhang, X. Hou, Y. Ren, and Y. Qian. Distributed q-learning aided heterogeneous network association for energy-efficient iiot. *IEEE Transactions on Industrial Informatics*, 16(4):2756–2764, 2020.

[148] L. Mai, N.-N. Dao, and M. Park. Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing. *Sensors (Switzerland)*, 18(9), 2018.

[149] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal*, 7(7):6360–6368, 2020.

[150] M. Khichane, P. Albert, and C. Solnon. Strong combination of ant colony optimization with constraint programming optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6140 LNCS:232–245, 2010.

[151] A. Villalonga, G. Beruvides, F. Castano, and R.E. Haber. Cloud-based industrial cyber-physical system for data-driven reasoning: A review and use case on an industry 4.0 pilot line. *IEEE Transactions on Industrial Informatics*, 16(9):5975–5984, 2020.

[152] T.-D. Lee, B.M. Lee, and W. Noh. Hierarchical cloud computing architecture for context-aware iot services. *IEEE Transactions on Consumer Electronics*, 64(2):222–230, 2018.

[153] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu. Multi-uav-enabled load-balance mobile-edge computing for iot networks. *IEEE Internet of Things Journal*, 7(8):6898–6908, 2020.

[154] X. Fu, F.R. Yu, J. Wang, Q. Qi, and J. Liao. Dynamic service function chain embedding for nfv-enabled iot: A deep reinforcement learning approach. *IEEE Transactions on Wireless Communications*, 19(1):507–519, 2020.

[155] X. Xiong, K. Zheng, L. Lei, and L. Hou. Resource allocation based on deep reinforcement learning in iot edge computing. *IEEE Journal on Selected Areas in Communications*, 38(6):1133–1146, 2020.

[156] A. Chowdhury, S.A. Raut, and H.S. Narman. Da-drls: Drift adaptive deep reinforcement learning based scheduling for iot resource management. *Journal of Network and Computer Applications*, 138:51–65, 2019.

[157] X. Fu, F.R. Yu, J. Wang, Q. Qi, and J. Liao. Service function chain embedding for nfv-enabled iot based on deep reinforcement learning. *IEEE Communications Magazine*, 57(11):102–108, 2019.

[158] F.M. Talaat, M.S. Saraya, A.I. Saleh, H.A. Ali, and S.H. Ali. A load balancing and optimization strategy (lbos) using reinforcement learning in fog computing environment. *Journal of Ambient Intelligence and Humanized Computing*, 11(11):4951–4966, 2020.

[159] C. Qiu, H. Yao, F.R. Yu, C. Jiang, and S. Guo. A service-oriented permissioned blockchain for the internet of things. *IEEE Transactions on Services Computing*, 13(2):203–215, 2020.

[160] J. Yao and N. Ansari. Task allocation in fog-aided mobile iot by lyapunov online reinforcement learning. *IEEE Transactions on Green Communications and Networking*, 4(2):556–565, 2020.

[161] R. Zhao, X. Wang, J. Xia, and L. Fan. Deep reinforcement learning based mobile edge computing for intelligent internet of things. *Physical Communication*, 43, 2020.

[162] M. Nduwayezu, Q.-V. Pham, and W.-J. Hwang. Online computation offloading in noma-based multi-access edge computing: A deep reinforcement learning approach. *IEEE Access*, 8:99098–99109, 2020.

[163] T. Fu, C. Wang, and N. Cheng. Deep-learning-based joint optimization of renewable energy storage and routing in vehicular energy network. *IEEE Internet of Things Journal*, 7(7):6229–6241, 2020.

[164] S. Guo, Y. Dai, S. Xu, X. Qiu, and F. Qi. Trusted cloud-edge network resource management: Drl-driven service function chain orchestration for iot. *IEEE Internet of Things Journal*, 7(7):6010–6022, 2020.

[165] Y. Zhang, B. Song, Y. Zhang, X. Du, and M. Guizani. Market model for resource allocation in emerging sensor networks with reinforcement learning. *Sensors (Switzerland)*, 16(12), 2016.

[166] S. Wan, J. Lu, P. Fan, and K.B. Letaief. Toward big data processing in iot: Path planning and resource management of uav base stations in mobile-edge computing system. *IEEE Internet of Things Journal*, 7(7):5995–6009, 2020.

[167] G. Cui, X. Li, L. Xu, and W. Wang. Latency and energy optimization for mec enhanced sat-iot networks. *IEEE Access*, 8:55915–55926, 2020.

[168] H. Yang, X. Xie, and M. Kadoch. Machine learning techniques and a case study for intelligent wireless networks. *IEEE Network*, 34(3):208–215, 2020.

[169] I. Khan, X. Tao, G.M. Shafiqur Rahman, W.U. Rehman, and T. Salam. Advanced energy-efficient computation offloading using deep reinforcement learning in mtc edge computing. *IEEE Access*, 8:82867–82875, 2020.

[170] H. Yang, W.-D. Zhong, C. Chen, A. Alphones, and X. Xie. Deep-reinforcement-learning-based energy-efficient resource management for social

and cognitive internet of things. *IEEE Internet of Things Journal*, 7(6):5677–5689, 2020.

[171] Q. Li, H. Yao, T. Mai, C. Jiang, and Y. Zhang. Reinforcement-learning-and belief-learning-based double auction mechanism for edge computing resource allocation. *IEEE Internet of Things Journal*, 7(7):5976–5985, 2020.

[172] Y. Liu, S. Xie, and Y. Zhang. Cooperative offloading and resource management for uav-enabled mobile edge computing in power iot system. *IEEE Transactions on Vehicular Technology*, 69(10):12229–12239, 2020.

[173] A. Ashiquzzaman, H. Lee, T.-W. Um, and J. Kim. Energy-efficient iot sensor calibration with deep reinforcement learning. *IEEE Access*, 8:97045–97055, 2020.

[174] J. Zhang, M. Dai, and Z. Su. Task allocation with unmanned surface vehicles in smart ocean iot. *IEEE Internet of Things Journal*, 7(10):9702–9713, 2020.

[175] D. Wang, W. Zhang, B. Song, X. Du, and M. Guizani. Market-based model in cr-iot: A q-probabilistic multi-agent reinforcement learning approach. *IEEE Transactions on Cognitive Communications and Networking*, 6(1):179–188, 2020.

[176] S. Xu, Q. Liu, B. Gong, F. Qi, S. Guo, X. Qiu, and C. Yang. Rjcc: Reinforcement-learning-based joint communicational-and-computational resource allocation mechanism for smart city iot. *IEEE Internet of Things Journal*, 7(9):8059–8076, 2020.

[177] X. Chen and G. Liu. Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks. *IEEE Internet of Things Journal*, 8(13):10843–10856, 2021.

[178] Y. Zhao, L. Wang, S. Li, F. Zhou, X. Lin, Q. Lu, and L. Ren. A visual analysis approach for understanding durability test data of automotive products. *ACM Transactions on Intelligent Systems and Technology*, 10(6), 2019.

[179] Y.-H. Xu, Y.-B. Tian, P.K. Searyoh, G. Yu, and Y.-T. Yong. Deep reinforcement learning-based resource allocation strategy for energy harvesting-powered cognitive machine-to-machine networks. *Computer Communications*, 160:706–717, 2020.

[180] N.N. Khumalo, O.O. Oyerinde, and L. Mfupe. Reinforcement learning-based resource management model for fog radio access network architectures in 5g. *IEEE Access*, 9:12706–12716, 2021.

[181] S. Ge, B. Lu, L. Xiao, J. Gong, X. Chen, and Y. Liu. Mobile edge computing against smart attacks with deep reinforcement learning in cognitive mimo iot systems. *Mobile Networks and Applications*, 25(5):1851–1862, 2020.

[182] I. Alqerm and J. Pan. Enhanced online q-learning scheme for resource allocation with maximum utility and fairness in edge-iot networks. *IEEE Transactions on Network Science and Engineering*, 7(4):3074–3086, 2020.

[183] Q. Qi, L. Zhang, J. Wang, H. Sun, Z. Zhuang, J. Liao, and F.R. Yu. Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(11):13861–13874, 2020.

[184] G. Sun, R. Ou, and G. Liu. Deep reinforcement learning-based resource reservation algorithm for emergency internet-of-things slice. *Tongxin Xuebao/Journal on Communications*, 41(9):8–20, 2020.

[185] Y. Liao, X. Qiao, Q. Yu, and Q. Liu. Intelligent dynamic service pricing strategy for multi-user vehicle-aided mec networks. *Future Generation Computer Systems*, 114:15–22, 2021.

[186] H. Qin, S. Zawad, Y. Zhou, S. Padhi, L. Yang, and F. Yan. Reinforcement-learning-empowered mlaas scheduling for serving intelligent internet of things. *IEEE Internet of Things Journal*, 7(7):6325–6337, 2020.

[187] S. Venticinque, S. Nacchia, and S.A. Maisto. Reinforcement learning for resource allocation in cloud datacenter. *Lecture Notes in Networks and Systems*, 96:648–657, 2020.

[188] V.K. Prasad and M.D. Bhavsar. Monitoring and prediction of sla for iot based cloud. *Scalable Computing*, 21(3):349–357, 2020.

[189] T. Liu, R. Luo, F. Xu, C. Fan, and C. Zhao. Distributed learning based joint communication and computation strategy of iot devices in smart cities. *Sensors (Switzerland)*, 20(4), 2020.

[190] I. Budhiraja, N. Kumar, and S. Tyagi. Deep-reinforcement-learning-based proportional fair scheduling control scheme for underlay d2d communication. *IEEE Internet of Things Journal*, 8(5):3143–3156, 2021.

[191] S. Ramakrishna, C. Harstell, M.P. Burruss, G. Karsai, and A. Dubey. Dynamic-weighted simplex strategy for learning enabled cyber physical systems. *Journal of Systems Architecture*, 111, 2020.

[192] M. Li, F.R. Yu, P. Si, W. Wu, and Y. Zhang. Resource optimization for delay-tolerant data in blockchain-enabled iot with edge computing: A deep reinforcement learning approach. *IEEE Internet of Things Journal*, 7(10):9399–9412, 2020.

[193] A. Sapio, S.S. Bhattacharyya, and M. Wolf. Runtime adaptation in wireless sensor nodes using structured learning. *ACM Transactions on Cyber-Physical Systems*, 4(4), 2020.

[194] K. Priyadharshini and R.A. Canessane. Light chain consensus reinforcement machine learning: An effective blockchain model for internet of things using for its advancement and challenges. *Computational Intelligence*, 36:1–22, 2020.

[195] N. Yuan, C. Jia, J. Lu, S. Guo, W. Li, X. Qiu, and L. Shi. A drl-based container placement scheme with auxiliary tasks. *Computers, Materials and Continua*, 64(3):1657–1671, 2020.

[196] M. Laroui, H. Ibn-Khedher, M. Ali Cherif, H. Moungla, H. Afifi, and A.E. Kamel. So-vmec: Service offloading in virtual mobile edge computing using deep reinforcement learning. *Transactions on Emerging Telecommunications Technologies*, 32, 2021.

[197] J. Kim, D. Ryu, J. Kim, and J.-H. Kim. Two-stage hybrid network clustering using multi-agent reinforcement learning. *Electronics (Switzerland)*, 10(3):1–16, 2021.

[198] X. Shu, L. Wu, X. Qin, R. Yang, Y. Wu, D. Wang, and B. Liao. Deep reinforcement learning cloud-edge-terminal computation resource allocation mechanism for iot. *Advances in Intelligent Systems and Computing*, 1274 AISC:1550–1556, 2021.

[199] J. Zhang, H. Guo, and J. Liu. Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme. *Mobile Networks and Applications*, 25(5):1736–1745, 2020.

[200] A.P. Ortega, S.D. Ramchurn, L. Tran-Thanh, and G.V. Merrett. Partner selection in self-organised wireless sensor networks for opportunistic energy negotiation: A multi-armed bandit based approach. *Ad Hoc Networks*, 112, 2021.

[201] H. Gao, Y. Xiao, H. Yan, Y. Tian, D. Wang, and W. Wang. A learning-based credible participant recruitment strategy for mobile crowd sensing. *IEEE Internet of Things Journal*, 7(6):5302–5314, 2020.

[202] H. Yang, Z. Xiong, J. Zhao, D. Niyato, C. Yuen, and R. Deng. Deep reinforcement learning based massive access management for ultra-reliable low-latency communications. *IEEE Transactions on Wireless Communications*, 20(5):2977–2990, 2021.

[203] Y. Liu, H. Lu, X. Li, Y. Zhang, L. Xi, and D. Zhao. Dynamic service function chain orchestration for nfv/mec-enabled iot networks: A deep reinforcement learning approach. *IEEE Internet of Things Journal*, 8(9):7450–7465, 2020.

[204] W. Zhang, D. Yang, P. Haixia, W. Wu, W. Quan, H. Zhang, and X. Shen. Deep reinforcement learning based resource management for dnn inference in industrial iot. *IEEE Transactions on Vehicular Technology*, 70(8):7605–7618, 2021.

[205] J. Zhu, Y. Song, D. Jiang, and H. Song. A new deep-q-learning-based transmission scheduling mechanism for the cognitive internet of things. *IEEE Internet of Things Journal*, 5(4):2375–2385, 2018.

[206] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang. Intelligent edge computing for iot-based energy management in smart cities. *IEEE Network*, 33(2):111–117, 2019.

[207] Y.-R. Shiue, K.-C. Lee, and C.-T. Su. Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers and Industrial Engineering*, 125:604–614, 2018.

[208] S.K. Sharma and X. Wang. Toward massive machine type communications in ultra-dense cellular iot networks: Current issues and machine learning-assisted solutions. *IEEE Communications Surveys and Tutorials*, 22(1):426–471, 2020.

[209] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, and X. Wang. Multiuser resource control with deep reinforcement learning in iot edge computing. *IEEE Internet of Things Journal*, 6(6):10119–10133, 2019.

[210] J. Ge, B. Liu, T. Wang, Q. Yang, A. Liu, and A. Li. Q-learning based flexible task scheduling in a global view for the internet of things. *Transactions on Emerging Telecommunications Technologies*, 32(8), 2021.

[211] Q. Tan, Y. Tong, S. Wu, and D. Li. Modeling, planning, and scheduling of shop-floor assembly process with dynamic cyber-physical interactions: a case study for cps-based smart industrial robot production. *International Journal of Advanced Manufacturing Technology*, 105(9):3979–3989, 2019.

[212] P. Gazori, D. Rahbari, and M. Nickray. Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach. *Future Generation Computer Systems*, 110:1098–1115, 2020.

[213] B. Yin, S. Zhang, and Y. Cheng. Application-oriented scheduling for optimizing the age of correlated information: A deep-reinforcement-learning-based approach. *IEEE Internet of Things Journal*, 7(9):8748–8759, 2020.

[214] D. Kim, T. Lee, S. Kim, B. Lee, and H.Y. Youn. Adaptive packet scheduling in iot environment based on q-learning. *Journal of Ambient Intelligence and Humanized Computing*, 11(6):2225–2235, 2020.

[215] S. Park, S. Park, M.-I. Choi, S. Lee, T. Lee, S. Kim, K. Cho, and S. Park. Reinforcement learning-based bems architecture for energy usage optimization. *Sensors (Switzerland)*, 20(17):1–33, 2020.

[216] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang. Edge-aided computing and transmission scheduling for lte-u-enabled iot. *IEEE Transactions on Wireless Communications*, 19(12):7881–7896, 2020.

[217] X. Fu, L. Lopez-Estrada, and J.G. Kim. A q-learning-based approach for enhancing energy efficiency of bluetooth low energy. *IEEE Access*, 9:21286–21295, 2021.

[218] H.-S. Lee and J.-W. Lee. Adaptive wireless power transfer beam scheduling for non-static iot devices using deep reinforcement learning. *IEEE Access*, 8:206659–206673, 2020.

[219] M. Samir, C. Assi, S. Sharafeddine, and A. Ghrayeb. Online altitude control and scheduling policy for minimizing aoi in uav-assisted iot wireless networks. *IEEE Transactions on Mobile Computing*, 19, 2020.

[220] H. Park, H. Kim, S.-T. Kim, and P. Mah. Multi-agent reinforcement-learning-based time-slotted channel hopping medium access control scheduling scheme. *IEEE Access*, 8:139727–139736, 2020.

[221] Y. Martínez Jiménez, J. Coto Palacio, and A. Nowé. Multi-agent reinforcement learning tool for job shop scheduling problems. *Communications in Computer and Information Science*, 1173 CCIS:3–12, 2020.

[222] H. Hu, X. Jia, Q. He, S. Fu, and K. Liu. Deep reinforcement learning based agvs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers and Industrial Engineering*, 149, 2020.

[223] H. Rashtian and S. Gopalakrishnan. Using deep reinforcement learning to improve sensor selection in the internet of things. *IEEE Access*, 8:95208–95222, 2020.

[224] Z. Wang, J. Wang, F. Yang, and M. Lin. Q-learning-based energy transmission scheduling over a fading channel. *Journal of Southeast University (English Edition)*, 36(4):393–398, 2020.

[225] N.C. Luong, D.T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D.I. Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys and Tutorials*, 21(4):3133–3174, 2019.

[226] T. Yu, B. Zhou, K.W. Chan, L. Chen, and B. Yang. Stochastic optimal relaxed automatic generation control in non-markov environment based on multi-step q(λ) learning. *IEEE Transactions on Power Systems*, 26(3):1272–1282, 2011.

[227] T. Yu, H.Z. Wang, B. Zhou, K.W. Chan, and J. Tang. Multi-agent correlated equilibrium q(λ) learning for coordinated smart generation control of interconnected power grids. *IEEE Transactions on Power Systems*, 30(4):1669–1679, 2015.

[228] T. Yu, B. Zhou, K.W. Chan, Y. Yuan, B. Yang, and Q.H. Wu. R(λ) imitation learning for automatic generation control of interconnected power grids. *Automatica*, 48(9):2130–2136, 2012.

[229] J. Xia, Y. Xu, D. Deng, Q. Zhou, and L. Fan. Intelligent secure communication for internet of things with statistical channel state information of attacker. *IEEE Access*, 7:144481–144488, 2019.

[230] J. Shao, X. Zhao, J. Yang, W. Zhang, Y. Kang, and X. Zhao. Reinforcement learning algorithm for path following control of articulated vehicle. *Nongye Jixie Xuebao/Transactions of the Chinese Society for Agricultural Machinery*, 48(3):376–382, 2017.

[231] H.-D. Tran, F. Cai, M. Lopez Diego, P. Musau, T.T. Johnson, and X. Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems*, 18(5s), 2019.

[232] J. Kang and D.-S. Eom. Offloading and transmission strategies for iot edge devices and networks. *Sensors (Switzerland)*, 19(4), 2019.

[233] X. Zhang, T. Yu, and J. Tang. Optimal cps command dispatch based on hierarchically correlated equilibrium reinforcement learning. *Dianli Xitong Zidonghua/Automation of Electric Power Systems*, 39(8):80–86, 2015.

[234] G. Faraci, A. Raciti, S.A. Rizzo, and G. Schembra. Green wireless power transfer system for a drone fleet managed by reinforcement learning in smart industry. *Applied Energy*, 259, 2020.

[235] J.-B. Kim, H.-K. Lim, C.-M. Kim, M.-S. Kim, Y.-G. Hong, and Y.-H. Han. Imitation reinforcement learning-based remote rotary inverted pendulum control in openflow network. *IEEE Access*, 7:36682–36690, 2019.

[236] D. Sikeridis, E.E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou. Energy-efficient orchestration in wireless powered internet of things infrastructures. *IEEE Transactions on Green Communications and Networking*, 3(2):317–328, 2019.

[237] B.-N. Trinh, L. Murphy, and G.-M. Muntean. A reinforcement learning-based duty cycle adjustment technique in wireless multimedia sensor networks. *IEEE Access*, 8:58774–58787, 2020.

[238] D. Pacheco-Paramo, L. Tello-Oquendo, V. Pla, and J. Martinez-Bauset. Deep reinforcement learning mechanism for dynamic access control in wireless networks handling mmtc. *Ad Hoc Networks*, 94, 2019.

[239] H.-K. Lim, J.-B. Kim, J.-S. Heo, and Y.-H. Han. Federated reinforcement learning for training control policies on multiple iot devices. *Sensors (Switzerland)*, 20(5), 2020.

[240] T.H.A. Kolobe and A.H. Fagg. Robot reinforcement and error-based movement learning in infants with and without cerebral palsy. *Physical Therapy*, 99(6):677–688, 2019.

[241] R. Ali, B. Kim, S.W. Kim, H.S. Kim, and F. Ishmanov. (relbt): A reinforcement learning-enabled listen before talk mechanism for lte-laa and wi-fi coexistence in iot. *Computer Communications*, 150:498–505, 2020.

[242] T. Yu and S.-P. Zhang. Automatic control of electricity generation based on 5-component update learning algorithm sarsa(λ). *Kongzhi Lilun Yu Yingyong/Control Theory and Applications*, 30(10):1246–1251, 2013.

[243] C. Liu, J. Gao, Y. Bi, X. Shi, and D. Tian. A multitasking-oriented robot arm motion planning scheme based on deep reinforcement learning and twin synchro-control. *Sensors (Switzerland)*, 20(12):1–35, 2020.

[244] L. An and G.-H. Yang. Opacity enforcement for confidential robust control in linear cyber-physical systems. *IEEE Transactions on Automatic Control*, 65(3):1234–1241, 2020.

[245] G. Faraci, C. Grasso, and G. Schembra. Fog in the clouds: Uavs to provide edge computing to iot devices. *ACM Transactions on Internet Technology*, 20(3), 2020.

[246] H. Joo, S.H. Ahmed, and Y. Lim. Traffic signal control for smart cities using reinforcement learning. *Computer Communications*, 154:324–330, 2020.

[247] Q. Wu, J. Wu, J. Shen, B. Yong, and Q. Zhou. An edge based multi-agent auto communication method for traffic light control. *Sensors (Switzerland)*, 20(15):1–16, 2020.

[248] T. Yu, S. Zhang, and Y. Hong. Dynamic optimal cps control for interconnected power systems based on sarsa algorithm. *Lecture Notes in Electrical Engineering*, 238 LNEE:269–276, 2014.

[249] H. Xu, X. Liu, W. Yu, D. Griffith, and N. Golmie. Reinforcement learning-based control and networking co-design for industrial internet of things. *IEEE Journal on Selected Areas in Communications*, 38(5):885–898, 2020.

[250] T. Wang, X. Shen, M.S. Obaidat, X. Liu, and S. Wan. Edge-learning-based hierarchical prefetching for collaborative information streaming in social iot systems. *IEEE Transactions on Computational Social Systems*, 7, 2020.

[251] S. Liu, S. Li, and B. Xu. Event-triggered resilient control for cyber-physical system under denial-of-service attacks. *International Journal of Control*, 93(8):1907–1919, 2020.

[252] S. Souihi, M. Souidi, and A. Mellouk. An adaptive qoe-based network interface selection for multi-homed ehealth devices. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 169:437–442, 2016.

[253] H. Van Dong, B. Quoc Khanh, N. Tran Lich, and N.T. Ngoc Anh. Integrating multi-agent system, geographic information system, and reinforcement learning to simulate and optimize traffic signal control. *Advances in Intelligent Systems and Computing*, 769:145–154, 2019.

[254] V. Hakami, S. Mostafavi, N.T. Javan, and Z. Rashidi. An optimal policy for joint compression and transmission control in delay-constrained energy harvesting iot devices. *Computer Communications*, 160:554–566, 2020.

[255] S. Kim. One-player game based influential maximization scheme for social cloud service networks. *EAI/Springer Innovations in Communication and Computing*, pages 175–184, 2019.

[256] D. Pacheco-Paramo and L. Tello-Oquendo. Delay-aware dynamic access control for mmtc in wireless networks using deep reinforcement learning. *Computer Networks*, 182, 2020.

[257] Y. Zhao, J. Hu, K. Yang, and S. Cui. Deep reinforcement learning aided intelligent access control in energy harvesting based wlan. *IEEE Transactions on Vehicular Technology*, 69(11):14078–14082, 2020.

[258] Y. Hadjadj-Aoul and S. Ait-Chellouche. Access control in nb-iot networks: A deep reinforcement learning strategy. *Information (Switzerland)*, 11(11):1–16, 2020.

[259] S. Khairy, P. Balaprakash, L.X. Cai, and Y. Cheng. Constrained deep reinforcement learning for energy sustainable multi-uav based random access iot networks with noma. *IEEE Journal on Selected Areas in Communications*, 39(4):1101–1115, 2021.

[260] M. Naeem, S.T.H. Rizvi, and A. Coronato. A gentle introduction to reinforcement learning and its application in different fields. *IEEE Access*, 8:209320–209344, 2020.

[261] F. Zhang, Q. Yang, and D. An. Cddpg: A deep-reinforcement-learning-based approach for electric vehicle charging control. *IEEE Internet of Things Journal*, 8(5):3075–3087, 2021.

[262] W. Wu, F. Zhu, Y.C. Fu, and Q. Liu. Deep deterministic policy gradient with clustered prioritized sampling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11302 LNCS:645–654, 2018.

[263] C. Cho, S. Shin, H. Jeon, and S. Yoon. Qos-aware workload distribution in hierarchical edge clouds: A reinforcement learning approach. *IEEE Access*, 8:193297–193313, 2020.

[264] A. Musaddiq, Z. Nain, Y.A. Qadri, R. Ali, and S.W. Kim. Reinforcement learning-enabled cross-layer optimization for low-power and lossy networks under heterogeneous traffic patterns. *Sensors (Switzerland)*, 20(15):1–25, 2020.

[265] R. Huang, V.W.S. Wong, and R. Schober. Throughput optimization for grant-free multiple access with multiagent deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 20(1):228–242, 2021.

[266] T. Yu and B. Zhou. Reinforcement learning based cps self-tuning control methodology for interconnected power systems. *Dianli Xitong Baohu yu Kongzhi/Power System Protection and Control*, 37(10):33–38, 2009.

[267] T. Yu and Y. Yuan. An average reward model based whole process r(λ)-learning for optimal cps control. *Dianli Xitong Zidonghua/Automation of Electric Power Systems*, 34(21):27–33, 2010.

[268] T. Liu, B. Tian, Y. Ai, and F.-Y. Wang. Parallel reinforcement learning-based energy efficiency improvement for a cyber-physical system. *IEEE/CAA Journal of Automatica Sinica*, 7(2):617–626, 2020.

[269] M. Kiermeier, S. Feld, T. Phan, and C. Linnhoff-Popien. Anomaly detection in spatial layer models of autonomous agents. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11314 LNCS:156–163, 2018.

[270] S. Jeong, G. Yoo, M. Yoo, I. Yeom, and H. Woo. Resource-efficient sensor data management for autonomous systems using deep reinforcement learning. *Sensors (Switzerland)*, 19(20), 2019.

[271] M. Kadohisa, J.V. Verhagen, and E.T. Rolls. The primate amygdala: Neuronal representations of the viscosity, fat texture, temperature, grittiness and taste of foods. *Neuroscience*, 132(1):33–48, 2005.

[272] C. Schwenck, A. Ciaramidaro, M. Selivanova, J. Tournay, C.M. Freitag, and M. Siniatchkin. Neural correlates of affective empathy and reinforcement learning in boys with conduct problems: fmri evidence from a gambling task. *Behavioural Brain Research*, 320:75–84, 2017.

[273] M. Li, W.-J. Liu, B. Lu, Y.-H. Wang, and J.-G. Liu. Differential expression of arc in the mesocorticolimbic system is involved in drug and natural rewarding behavior in rats. *Acta Pharmacologica Sinica*, 34(8):1013–1024, 2013.

[274] C. Wang, J. Wang, J. Wang, and X. Zhang. Deep-reinforcement-learning-based autonomous uav navigation with sparse rewards. *IEEE Internet of Things Journal*, 7(7):6180–6190, 2020.

[275] H.-Y. Kim and J. Kim. A load balancing scheme for gaming server applying reinforcement learning in iot. *Computer Science and Information Systems*, 17(3):891–906, 2020.

[276] A.H. Keyhanipour, B. Moshiri, M. Rahgozar, F. Oroumchian, and A.A. Ansari. Integration of data fusion and reinforcement learning techniques for the rank-aggregation problem. *International Journal of Machine Learning and Cybernetics*, 7(6):1131–1145, 2016.

[277] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang. Learning-based computation offloading for iot devices with energy harvesting. *IEEE Transactions on Vehicular Technology*, 68(2):1930–1941, 2019.

[278] M. Min, X. Wan, L. Xiao, Y. Chen, M. Xia, D. Wu, and H. Dai. Learning-based privacy-aware offloading for healthcare iot with energy harvesting. *IEEE Internet of Things Journal*, 6(3):4307–4316, 2019.

[279] D. Sikeridis, E.E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou. Wireless powered public safety iot: A uav-assisted adaptive-learning approach towards energy efficiency. *Journal of Network and Computer Applications*, 123:69–79, 2018.

[280] Y. Cui, D. Zhang, T. Zhang, L. Chen, M. Piao, and H. Zhu. Novel method of mobile edge computation offloading based on evolutionary game strategy for iot devices. *AEU - International Journal of Electronics and Communications*, 118, 2020.

[281] K. Gai, K. Xu, Z. Lu, M. Qiu, and L. Zhu. Fusion of cognitive wireless networks and edge computing. *IEEE Wireless Communications*, 26(3):69–75, 2019.

[282] S. Spanò, G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Matta, A. Nannarelli, and M. Re. An efficient hardware implementation of reinforcement learning: The q-learning algorithm. *IEEE Access*, 7:186340–186351, 2019.

[283] M.S. Munir, S.F. Abedin, N.H. Tran, and C.S. Hong. When edge computing meets microgrid: A deep reinforcement learning approach. *IEEE Internet of Things Journal*, 6(5):7360–7374, 2019.

[284] N. Shoeibi and N. Shoeibi. Future of smart parking: Automated valet parking using deep q-learning. *Advances in Intelligent Systems and Computing*, 1004:177–182, 2020.

[285] Z. Wang, Y. Liu, Z. Ma, X. Liu, and J. Ma. Lipsg: Lightweight privacy-preserving q-learning-based energy management for the iot-enabled smart grid. *IEEE Internet of Things Journal*, 7(5):3935–3947, 2020.

[286] M. Ozturk, M. Jaber, and M.A. Imran. Energy-aware smart connectivity for iot networks: Enabling smart ports. *Wireless Communications and Mobile Computing*, 2018, 2018.

[287] H. Lu, X. He, M. Du, X. Ruan, Y. Sun, and K. Wang. Edge qoe: Computation offloading with deep reinforcement learning for internet of things. *IEEE Internet of Things Journal*, 7(10):9255–9265, 2020.

[288] D. Ma, G. Lan, M. Hassan, W. Hu, and S.K. Das. Sensing, computing, and communications for energy harvesting iots: A survey. *IEEE Communications Surveys and Tutorials*, 22(2):1222–1250, 2020.

[289] R. Bonnefoi, C. Moy, and J. Palicot. Improvement of the lpwan ami backhaul's latency thanks to reinforcement learning algorithms. *Eurasip Journal on Wireless Communications and Networking*, 2018(1):1–18, 2018.

[290] X. Bao, H. Liang, and L. Han. Transmission optimization of social and physical sensor nodes via collaborative beamforming in cyber-physical-social systems. *Sensors (Switzerland)*, 18(12), 2018.

[291] M. Han, J. Duan, S. Khairy, and L.X. Cai. Enabling sustainable underwater iot networks with energy harvesting: A decentralized reinforcement learning approach. *IEEE Internet of Things Journal*, 7(10):9953–9964, 2020.

[292] T. Mohammed, A. Albeshri, I. Katib, and R. Mehmood. Ubipriseq—deep reinforcement learning to manage privacy, security, energy, and qos in 5g iot hetnets. *Applied Sciences (Switzerland)*, 10(20):1–18, 2020.

[293] J. Tang, H. Tang, X. Zhang, K. Cumanan, G. Chen, K.-K. Wong, and J.A. Chambers. Energy minimization in d2d-assisted cache-enabled internet of things: A deep reinforcement learning approach. *IEEE Transactions on Industrial Informatics*, 16(8):5412–5423, 2020.

[294] G. Maselli, M. Piva, and J.A. Stankovic. Adaptive communication for battery-free devices in smart homes. *IEEE Internet of Things Journal*, 6(4):6977–6988, 2019.

[295] H. Ke, J. Wang, H. Wang, and Y. Ge. Joint optimization of data offloading and resource allocation with renewable energy aware for iot devices: A deep reinforcement learning approach. *IEEE Access*, 7:179349–179363, 2019.

[296] Y. Xie, Z. Xu, J. Xu, S. Gong, and Y. Wang. Backscatter-aided hybrid data offloading for mobile edge computing via deep reinforcement learning. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 294 LNCIST:525–537, 2019.

[297] Y. Rioual, J. Laurent, and J.-P. Diguet. Reinforcement-learning approach guidelines for energy management. *Journal of Low Power Electronics*, 15(3):283–293, 2019.

[298] C. Han, A. Liu, H. Wang, L. Huo, and X. Liang. Dynamic anti-jamming coalition for satellite-enabled army iot: A distributed game approach. *IEEE Internet of Things Journal*, 7(11):10932–10944, 2020.

[299] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang. Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale mec networks. *IEEE Internet of Things Journal*, 7(10):9278–9290, 2020.

[300] X. Tu, C. Xu, S. Liu, R. Li, G. Xie, J. Huang, and L.T. Yang. Efficient monocular depth estimation for edge devices in internet of things. *IEEE Transactions on Industrial Informatics*, 17(4):2821–2832, 2021.

[301] J. Long, Y. Luo, X. Zhu, E. Luo, and M. Huang. Computation offloading through mobile vehicles in iot-edge-cloud network. *Eurasip Journal on Wireless Communications and Networking*, 2020(1), 2020.

[302] Y. Akbari and S. Tabatabaei. A new method to find a high reliable route in iot by using reinforcement learning and fuzzy logic. *Wireless Personal Communications*, 112(2):967–983, 2020.

[303] Y. Li, X. Zhao, and H. Liang. Throughput maximization by deep reinforcement learning with energy cooperation for renewable ultradense iot networks. *IEEE Internet of Things Journal*, 7(9):9091–9102, 2020.

[304] J. Zheng, L. Gao, H. Wang, J. Niu, J. Ren, H. Guo, X. Yang, and Y. Liu. Smart edge caching-aided partial opportunistic interference alignment in hetnets. *Mobile Networks and Applications*, 25(5):1842–1850, 2020.

[305] M. Peng, S. Garg, X. Wang, A. Bradai, H. Lin, and M.S. Hossain. Learning-based iot data aggregation for disaster scenarios. *IEEE Access*, 8:128490–128497, 2020.

[306] S. Sarwar, R. Sirhindi, L. Aslam, G. Mustafa, M.M. Yousaf, and S.W.U.Q. Jaffry. Reinforcement learning based adaptive duty cycling in lr-wpans. *IEEE Access*, 8:161157–161174, 2020.

[307] K. Wang, C.-M. Chen, M.S. Hossain, G. Muhammad, S. Kumar, and S. Kumari. Transfer reinforcement learning-based road object detection in next generation iot domain. *Computer Networks*, 193, 2021.

[308] M.I. Khan, L. Reggiani, M.M. Alam, Y.L. Moullec, N. Sharma, E. Yaacoub, and M. Magarini. Q-learning based joint energy-spectral efficiency optimization in multi-hop device-to-device communication. *Sensors (Switzerland)*, 20(22):1–23, 2020.

[309] S.F. Abedin, M.S. Munir, N.H. Tran, Z. Han, and C.S. Hong. Data freshness and energy-efficient uav navigation optimization: A deep reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5994–6006, 2021.

[310] G. Kaur, P. Chanak, and M. Bhattacharya. Energy efficient intelligent routing scheme for iot-enabled wsns. *IEEE Internet of Things Journal*, 8(14):11440–11449, 2021.

[311] Z. Xiong, Y. Zhang, W.Y.B. Lim, J. Kang, D. Niyato, C. Leung, and C. Miao. Uav-assisted wireless energy and data transfer with deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):85–99, 2021.

[312] Y. Nie, J. Zhao, J. Liu, J. Jiang, and R. Ding. Energy-efficient uav trajectory design for backscatter communication: A deep reinforcement learning approach. *China Communications*, 17(10):129–141, 2020.

[313] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah. Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Communications Surveys and Tutorials*, 21(4):3039–3071, 2019.

[314] F.M. Al-Turjman. Information-centric sensor networks for cognitive iot: an overview. *Annales des Telecommunications/Annals of Telecommunications*, 72(1-2):3–18, 2017.

[315] Y. Liu, H. Yu, S. Xie, and Y. Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, 68(11):11158–11168, 2019.

[316] H. Khelifi, S. Luo, B. Nour, A. Sellami, H. Moungla, S.H. Ahmed, and M. Guizani. Bringing deep learning at the edge of information-centric internet of things. *IEEE Communications Letters*, 23(1):52–55, 2019.

[317] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia. Machine learning for wireless communications in the internet of things: A comprehensive survey. *Ad Hoc Networks*, 93, 2019.

[318] R.M. Sandoval, A.-J. Garcia-Sanchez, and J. Garcia-Haro. Optimizing and updating lora communication parameters: A machine learning approach. *IEEE Transactions on Network and Service Management*, 16(3):884–895, 2019.

[319] H. Song, J. Bai, Y. Yi, J. Wu, and L. Liu. Artificial intelligence enabled internet of things: Network architecture and spectrum access. *IEEE Computational Intelligence Magazine*, 15(1):44–51, 2020.

[320] A. Foerster, A. Udugama, C. Görg, K. Kuladinithi, A. Timm-Giel, and A. Cama-Pinto. A novel data dissemination model for organic data flows. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 158:239–252, 2015.

[321] S. Shukla, M.F. Hassan, M.K. Khan, L.T. Jung, and A. Awang. An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. *PLoS ONE*, 14(11), 2019.

[322] A. Asheralieva and D. Niyato. Distributed dynamic resource management and pricing in the iot systems with blockchain-as-a-service and uav-enabled

mobile edge computing. *IEEE Internet of Things Journal*, 7(3):1974–1993, 2020.

[323] R. Bajracharya, R. Shrestha, and S.W. Kim. Q-learning based fair and efficient coexistence of lte in unlicensed band. *Sensors (Switzerland)*, 19(13), 2019.

[324] X. Guo, H. Lin, Z. Li, and M. Peng. Deep-reinforcement-learning-based qos-aware secure routing for sdn-iot. *IEEE Internet of Things Journal*, 7(7):6242–6251, 2020.

[325] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu. Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0. *IEEE Internet of Things Journal*, 7(7):6201–6213, 2020.

[326] M. McClellan, C. Cervelló-Pastor, and S. Sallent. Deep learning at the mobile edge: Opportunities for 5g networks. *Applied Sciences (Switzerland)*, 10(14), 2020.

[327] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li. Collaborative learning of communication routes in edge-enabled multi-access vehicular environment. *IEEE Transactions on Cognitive Communications and Networking*, 6(4):1155–1165, 2020.

[328] A. Serhani, N. Naja, and A. Jamali. Aq-routing: mobility-, stability-aware adaptive routing protocol for data routing in manet–iot systems. *Cluster Computing*, 23(1):13–27, 2020.

[329] F. Jameel, U. Javaid, W.U. Khan, M.N. Aman, H. Pervaiz, and R. Jäntti. Reinforcement learning in blockchain-enabled iiot networks: A survey of recent advances and open challenges. *Sustainability (Switzerland)*, 12(12), 2020.

[330] J.M.C. Neto, S.F.G. Neto, P.M. de Santana, and V.A. de Sousa. Multi-cell lte-u/wi-fi coexistence evaluation using a reinforcement learning framework. *Sensors (Switzerland)*, 20(7), 2020.

[331] W. Ejaz, M. Basharat, S. Saadat, A.M. Khattak, M. Naeem, and A. Anpalagan. Learning paradigms for communication and computing technologies in iot systems. *Computer Communications*, 153:11–25, 2020.

[332] J. Tang, J. Song, J. Ou, J. Luo, X. Zhang, and K.-K. Wong. Minimum throughput maximization for multi-uav enabled wpcn: A deep reinforcement learning method. *IEEE Access*, 8:9124–9132, 2020.

[333] A. Abane, M. Daoui, S. Bouzefrane, and P. Muhlethaler. A lightweight forwarding strategy for named data networking in low-end iot. *Journal of Network and Computer Applications*, 148, 2019.

[334] R. Ali, Y.B. Zikria, B.-S. Kim, and S.W. Kim. Deep reinforcement learning paradigm for dense wireless networks in smart cities. *EAI/Springer Innovations in Communication and Computing*, pages 43–70, 2020.

[335] Q. Zhang, Y.-C. Liang, and H.V. Poor. Intelligent user association for symbiotic radio networks using deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 19(7):4535–4548, 2020.

[336] R.M. Sandoval, S. Canovas-Carrasco, A.-J. Garcia-Sanchez, and J. Garcia-Haro. A reinforcement learning-based framework for the exploitation of multiple rats in the iot. *IEEE Access*, 7:123341–123354, 2019.

[337] C. Sun, H. Ding, and X. Liu. Multichannel spectrum access based on reinforcement learning in cognitive internet of things. *Ad Hoc Networks*, 106, 2020.

[338] G.L. Santos, P.T. Endo, D. Sadok, and J. Kelner. When 5g meets deep learning: A systematic review. *Algorithms*, 13(9), 2020.

[339] N. Chen, T. Qiu, C. Mu, M. Han, and P. Zhou. Deep actor-critic learning-based robustness enhancement of internet of things. *IEEE Internet of Things Journal*, 7(7):6191–6200, 2020.

[340] Y. Zhang, B. Feng, W. Quan, A. Tian, K. Sood, Y. Lin, and H. Zhang. Cooperative edge caching: A multi-agent deep learning based approach. *IEEE Access*, 8:133212–133224, 2020.

[341] Y. Hao, M. Li, D. Wu, M. Chen, M.M. Hassan, and G. Fortino. Human-like hybrid caching in software-defined edge cloud. *IEEE Internet of Things Journal*, 7(7):5806–5815, 2020.

[342] F. Dou, J. Lu, T. Xu, C.-H. Huang, and J. Bi. A bisection reinforcement learning approach to 3-d indoor localization. *IEEE Internet of Things Journal*, 8(8):6519–6535, 2021.

[343] T.-W. Ban. An autonomous transmission scheme using dueling dqn for d2d communication networks. *IEEE Transactions on Vehicular Technology*, 69(12):16348–16352, 2020.

[344] K.-S. Shin, G.-H. Hwang, and O. Jo. Distributed reinforcement learning scheme for environmentally adaptive iot network selection. *Electronics Letters*, 56(9):441–444, 2020.

[345] N. Garg, M. Sellathurai, V. Bhatia, and T. Ratnarajah. Function approximation based reinforcement learning for edge caching in massive mimo networks. *IEEE Transactions on Communications*, 69(4):2304–2316, 2021.

[346] N. Jiang, Y. Deng, A. Nallanathan, and J.A. Chambers. Reinforcement learning for real-time optimization in nb-iot networks. *IEEE Journal on Selected Areas in Communications*, 37(6):1424–1440, 2019.

[347] T.S.P. Kumar and P.V. Krishna. Power modelling of sensors for iot using reinforcement learning. *International Journal of Advanced Intelligence Paradigms*, 10(1-2):3–22, 2018.

[348] Y. Li, X. Hu, Y. Zhuang, Z. Gao, P. Zhang, and N. El-Sheimy. Deep reinforcement learning (drl): Another perspective for unsupervised wireless localization. *IEEE Internet of Things Journal*, 7(7):6279–6287, 2020.

[349] M. Li, C. Chen, C. Hua, and X. Guan. Intelligent latency-aware virtual network embedding for industrial wireless networks. *IEEE Internet of Things Journal*, 6(5):7484–7496, 2019.

[350] G.M. Dias, C.B. Margi, F.C.P. De Oliveira, and B. Bellalta. Cloud-empowered, self-managing wireless sensor networks: Interconnecting management operations at the application layer. *IEEE Consumer Electronics Magazine*, 8(1):55–60, 2019.

[351] C. Yang, K.-W. Chin, T. He, and Y. Liu. On sampling time maximization in wireless powered internet of things. *IEEE Transactions on Green Communications and Networking*, 3(3):641–650, 2019.

[352] A. Kawewong, Y. Honda, M. Tsuboyama, and O. Hasegawa. A common-neural-pattern based reasoning for mobile robot cognitive mapping. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5506 LNCS(PART 1):32–39, 2009.

[353] C. Moy, L. Besson, G. Delbarre, and L. Toutain. Decentralized spectrum learning for radio collision mitigation in ultra-dense iot networks: Lorawan case study and experiments. *Annales des Telecommunications/Annals of Telecommunications*, 75(11-12):711–727, 2020.

[354] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin. Caching transient data for internet of things: A deep reinforcement learning approach. *IEEE Internet of Things Journal*, 6(2):2074–2083, 2019.

[355] Z. Li, Y. Lu, Y. Shi, Z. Wang, W. Qiao, and Y. Liu. A dyna-q-based solution for uav networks against smart jamming attacks. *Symmetry*, 11(5), 2019.

[356] J. Luo, S. Green, P. Feghali, G. Legrady, and C.K. Koç. *Reinforcement learning and trustworthy autonomy.* Springer International Publishing, 2018.

[357] Y. Liu, W. Zhang, S. Pan, Y. Li, and Y. Chen. Analyzing the robotic behavior in a smart city with deep enforcement and imitation learning using iort. *Computer Communications*, 150:346–356, 2020.

[358] K. Watanabe and S. Inada. Search algorithm of the assembly sequence of products by using past learning results. *International Journal of Production Economics*, 226, 2020.

[359] K.D. Baek and I.-Y. Ko. Effect-driven selection of web of things services in cyber-physical systems using reinforcement learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11496 LNCS:554–559, 2019.

[360] I. Verner, M. Reitman, D. Cuperman, T. Yan, E. Finkelstein, and T. Romm. Exposing robot learning to students in augmented reality experience. *Lecture Notes in Networks and Systems*, 47:610–619, 2019.

[361] Y.-T. Tsai, C.-H. Lee, T.-Y. Liu, T.-J. Chang, C.-S. Wang, S.J. Pawar, P.-H. Huang, and J.-H. Huang. Utilization of a reinforcement learning algorithm for the accurate alignment of a robotic arm in a complete soft fabric shoe tongues automation process. *Journal of Manufacturing Systems*, 56:501–513, 2020.

[362] G. Serin, B. Sener, A.M. Ozbayoglu, and H.O. Unver. Review of tool condition monitoring in machining and opportunities for deep learning. *International Journal of Advanced Manufacturing Technology*, 109(3-4):953–974, 2020.

[363] D.B. Noureddine, M. Krichen, S. Mechti, T. Nahhal, and W.Y.H. Adoni. An agent-based architecture using deep reinforcement learning for the intelligent internet of things applications. *Advances in Intelligent Systems and Computing*, 1188:273–283, 2021.

[364] R.S. Alonso. Deep tech and artificial intelligence for worker safety in robotic manufacturing environments. *Advances in Intelligent Systems and Computing*, 1242 AISC:234–240, 2021.

[365] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh. Semisupervised deep reinforcement learning in support of iot and smart city services. *IEEE Internet of Things Journal*, 5(2):624–635, 2018.

[366] X. Zhang, L. Yao, S. Zhang, S. Kanhere, M. Sheng, and Y. Liu. Internet of things meets brain-computer interface: A unified deep learning framework for enabling human-thing cognitive interactivity. *IEEE Internet of Things Journal*, 6(2):2084–2092, 2019.

[367] O.A. Sianaki, A. Yousefi, A.R. Tabesh, and M. Mahdavi. Machine learning applications: The past and current research trend in diverse industries. *Inventions*, 4(1), 2019.

[368] G. Neelakantam, D.D. Onthoni, and P.K. Sahoo. Reinforcement learning based passengers assistance system for crowded public transportation in fog enabled smart city. *Electronics (Switzerland)*, 9(9):1–19, 2020.

[369] M. Rivas and F. Giorno. A reinforcement learning multiagent architecture prototype for smart homes (iot). *Advances in Intelligent Systems and Computing*, 880:159–170, 2019.

[370] N. Magaia, R. Fonseca, K. Muhammad, A.H.F.N. Segundo, A.V. Lira Neto, and V.H.C. De Albuquerque. Industrial internet-of-things security enhanced with deep learning approaches for smart cities. *IEEE Internet of Things Journal*, 8(8):6393–6405, 2021.

[371] S. Pan, P. Li, D. Zeng, S. Guo, and G. Hu. A q -learning based framework for congested link identification. *IEEE Internet of Things Journal*, 6(6):9668–9678, 2019.

[372] H. Lin, Z. Chen, and L. Wang. Offloading for edge computing in low power wide area networks with energy harvesting. *IEEE Access*, 7:78919–78929, 2019.

[373] D.K. Sharma, J.J.P.C. Rodrigues, V. Vashishth, A. Khanna, and A. Chhabra. Rlproph: a dynamic programming based reinforcement learning

approach for optimal routing in opportunistic iot networks. *Wireless Networks*, 26(6):4319–4338, 2020.

[374] B. Song, J. Song, and J. Ye. A dynamic pricing mechanism in iot for daas: A reinforcement learning approach. *Advances in Intelligent Systems and Computing*, 1075:604–615, 2020.

[375] D. Wang, X. Tian, H. Cui, and Z. Liu. Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware mec network. *China Communications*, 17(8):31–44, 2020.

[376] W. Shafik, S. Mojtaba Matinkhah, P. Etemadinejad, and M.N. Sanda. Reinforcement learning rebirth, techniques, challenges, and resolutions. *International Journal on Informatics Visualization*, 4(3):127–135, 2020.

[377] E. Erdemir, P.L. Dragotti, and D. Gunduz. Privacy-aware time-series data sharing with deep reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 16:389–401, 2021.

[378] P. Wang, L.T. Yang, J. Li, X. Li, and X. Zhou. Mmdp: A mobile-iot based multi-modal reinforcement learning service framework. *IEEE Transactions on Services Computing*, 13(4):675–684, 2020.

[379] W. Jiang, G. Feng, S. Qin, and Y. Liu. Multi-agent reinforcement learning based cooperative content caching for mobile edge networks. *IEEE Access*, 7:61856–61867, 2019.

[380] J. Ma, S. Hasegawa, S.-J. Kim, and M. Hasegawa. A reinforcement-learning-based distributed resource selection algorithm for massive iot. *Applied Sciences (Switzerland)*, 9(18), 2019.

[381] Y. Qian, L. Shi, J. Li, Z. Wang, H. Guan, F. Shu, and H.V. Poor. A workflow-aided internet of things paradigm with intelligent edge computing. *IEEE Network*, 34(6):92–99, 2020.

[382] Z. Shi, Y. Zeng, and Z. Wu. Service chain orchestration based on deep reinforcement learning in intent-based iot. *Advances in Intelligent Systems and Computing*, 1143:875–882, 2021.

[383] W.-C. Chien, H.-Y. Weng, and C.-F. Lai. Q-learning based collaborative cache allocation in mobile edge computing. *Future Generation Computer Systems*, 102:603–610, 2020.

[384] V. Vijayaraghavan and J.R. Leevinson. *Intelligent traffic management systems for next generation IoV in smart city scenario*. Springer International Publishing, 2020.

[385] T. Lee, O. Jo, and K. Shin. Corl: Collaborative reinforcement learning-based mac protocol for iot networks. *Electronics (Switzerland)*, 9(1), 2020.

[386] C. Kim. Deep reinforcement learning by balancing offline monte carlo and online temporal difference use based on environment experiences. *Symmetry*, 12(10):1–16, 2020.

[387] S. Misra, P.K. Deb, N. Koppala, A. Mukherjee, and S. Mao. S-nav: Safety-aware iot navigation tool for avoiding covid-19 hotspots. *IEEE Internet of Things Journal*, 8(8):6975–6982, 2021.

[388] D.N. Doan, D. Zaharie, and D. Petcu. Auto-scaling for a streaming architecture with fuzzy deep reinforcement learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11997 LNCS:476–488, 2020.

[389] Y. Liu, H. Wang, M. Peng, J. Guan, and Y. Wang. An incentive mechanism for privacy-preserving crowdsensing via deep reinforcement learning. *IEEE Internet of Things Journal*, 8(10):8616–8631, 2021.

[390] H. Guo, S. Li, B. Li, Y. Ma, and X. Ren. A new learning automata-based pruning method to train deep neural networks. *IEEE Internet of Things Journal*, 5(5):3263–3269, 2018.

[391] Y. Wang, X. Chen, L. Wang, and G. Min. Effective iot-facilitated storm surge flood modeling based on deep reinforcement learning. *IEEE Internet of Things Journal*, 7(7):6338–6347, 2020.

[392] J. Yun, Y. Goh, and J.-M. Chung. Dqn-based optimization framework for secure sharded blockchain systems. *IEEE Internet of Things Journal*, 8(2):708–722, 2021.

[393] K.E. Mwangi, S. Masupe, and J. Mandu. Modelling malware propagation on the internet of things using an agent-based approach on complex networks. *Jordanian Journal of Computers and Information Technology*, 6(1):26–40, 2020.

[394] X. He, K. Wang, and W. Xu. Qoe-driven content-centric caching with deep reinforcement learning in edge-enabled iot. *IEEE Computational Intelligence Magazine*, 14(4):12–20, 2019.

[395] T.G. Nguyen, T.V. Phan, D.T. Hoang, T.N. Nguyen, and C. So-In. Efficient sdn-based traffic monitoring in iot networks with double deep q-network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12575 LNCS:26–38, 2020.

[396] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian. Resource trading in blockchain-based industrial internet of things. *IEEE Transactions on Industrial Informatics*, 15(6):3602–3609, 2019.

[397] B. Banerjee and L. Kraemer. Action discovery for single and multi-agent reinforcement learning. *Advances in Complex Systems*, 14(2):279–305, 2011.

[398] Y. Li, F. Qi, Z. Wang, X. Yu, and S. Shao. Distributed edge computing offloading algorithm based on deep reinforcement learning. *IEEE Access*, 8:85204–85215, 2020.

[399] L. Zhou, Q. Liu, F. Wu, and Y. Wei. Deep learning based sensing resource allocation for mobile target tracking. volume 2020-October, pages 430–435. Institute of Electrical and Electronics Engineers Inc., 2020.

[400] P. Zhang, Y. Yuan, Z. Wang, and C. Sun. A hierarchical game approach to the coupled resilient control of cps against denial-of-service attack. volume 2019-July, pages 15–20. IEEE Computer Society, 2019.

[401] E. Loiseau, L. Saikku, R. Antikainen, N. Droste, B. Hansjürgens, K. Pitkänen, P. Leskinen, P. Kuikman, and M. Thomsen. Green economy and related concepts: An overview. *Journal of Cleaner Production*, 139:361–371, 2016.

[402] J. Camacho-Otero, C. Boks, and I.N. Pettersen. Consumption in the circular economy: A literature review. *Sustainability (Switzerland)*, 10(8), 2018.

[403] Y. Kalmykova, M. Sadagopan, and L. Rosado. Circular economy - from review of theories and practices to development of implementation tools. *Resources, Conservation and Recycling*, 135:190–201, 2018.

[404] B.M. Beamon. Measuring supply chain performance. *International Journal of Operations and Production Management*, 19(3):275–292, 1999.

[405] R. de Koster, T. Le-Duc, and K.J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501, 2007.

[406] A.A. Hervani, M.M. Helms, and J. Sarkis. Performance measurement for green supply chain management. *Benchmarking*, 12(4):330–353, 2005.

[407] F. Jovane, L. Alting, A. Armillotta, W. Eversheim, K. Feldmann, G. Seliger, and N. Roth. A key issue in product life cycle: Disassembly. *CIRP Annals - Manufacturing Technology*, 42(2):651–658, 1993.

[408] J.R. Duflou, G. Seliger, S. Kara, Y. Umeda, A. Ometto, and B. Willems. Efficiency and feasibility of product disassembly: A case-based study. *CIRP Annals - Manufacturing Technology*, 57(2):583–600, 2008.

[409] P. Sasikumar and G. Kannan. Issues in reverse supply chains, part i: End-of-life product recovery and inventory management - an overview. *International Journal of Sustainable Engineering*, 1(3):154–172, 2008.

[410] S.M. Gupta and C.R. McLean. Disassembly of products. *Computers and Industrial Engineering*, 31(1-2):225–228, 1996.

[411] H-J Kim, D-H Lee, and P Xirouchakis. Disassembly scheduling: literature review and future research directions. *International Journal of Production Research*, 45(18-19):4465–4484, 2007.

[412] Mirothali Chand and Chandrasekar Ravi. A state-of-the-art literature survey on artificial intelligence techniques for disassembly sequence planning. *CIRP Journal of Manufacturing Science and Technology*, 41:292–310, 2023.

[413] SM Gupta and KN Taleb. Scheduling disassembly. *The International Journal of Production Research*, 32(8):1857–1866, 1994.

[414] DH Lee, HJ Kim, G Choi, and P Xirouchakis. Disassembly scheduling: integer programming models. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 218(10):1357–1372, 2004.

[415] Hwa-Joong Kim, Dong-Ho Lee, P Xirouchakis, and OK Kwon. A branch and bound algorithm for disassembly scheduling with assembly product structure. *Journal of the Operational Research Society*, 60(3):419–430, 2009.

[416] Karim N Taleb, Surendra M Gupta, and Louis Brennan. Disassembly of complex product structures with parts and materials commonality. *Production Planning & Control*, 8(3):255–269, 1997.

[417] Karim N Taleb and Surendra M Gupta. Disassembly of multiple product structures. *Computers & Industrial Engineering*, 32(4):949–961, 1997.

[418] Klaus-Peter Neuendorf, Dong-Ho Lee, Dimitris Kiritsis, and Paul Xirouchakis. Disassembly scheduling with parts commonality using petri nets with timestamps. *Fundamenta Informaticae*, 47(3-4):295–306, 2001.

[419] Hui Wang, Yiming Rong, and Dong Xiang. Mechanical assembly planning using ant colony optimization. *Computer-Aided Design*, 47:59–71, 2014.

[420] JinFeng Wang, Xuehua Wu, and Xiaoliang Fan. A two-stage ant colony optimization approach based on a directed graph for process planning. *The International Journal of Advanced Manufacturing Technology*, 80(5):839–850, 2015.

[421] Yongtao Luo, Qingjin Peng, and Peihua Gu. Integrated multi-layer representation and ant colony search for product selective disassembly planning. *Computers in Industry*, 75:13–26, 2016.

[422] Hwai-En Tseng, Chien-Cheng Chang, Shih-Chen Lee, and Yu-Ming Huang. Hybrid bidirectional ant colony optimization (hybrid baco): an algorithm for disassembly sequence planning. *Engineering Applications of Artificial Intelligence*, 83:45–56, 2019.

[423] Yaping Ren, Guangdong Tian, Fu Zhao, Daoyuan Yu, and Chaoyong Zhang. Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm. *Engineering Applications of Artificial Intelligence*, 64:415–431, 2017.

[424] Guangdong Tian, Yaping Ren, Yixiong Feng, MengChu Zhou, Honghao Zhang, and Jianrong Tan. Modeling and planning for dual-objective selective disassembly using and/or graph and discrete artificial bee colony. *IEEE Transactions on Industrial Informatics*, 15(4):2456–2468, 2018.

[425] Wenjun Xu, Quan Tang, Jiayi Liu, Zhihao Liu, Zude Zhou, and Duc Truong Pham. Disassembly sequence planning using discrete bees algorithm for human-robot collaboration in remanufacturing. *Robotics and Computer-Integrated Manufacturing*, 62:101860, 2020.

[426] TF Go, DA Wahab, MN Ab Rahman, R Ramli, et al. A design framework for end-of-life vehicles recovery: optimization of disassembly sequence using genetic algorithms. *American Journal of Environmental Sciences*, 6(4):350, 2010.

[427] Jeremy L Rickli and Jaime A Camelio. Multi-objective partial disassembly optimization based on sequence feasibility. *Journal of manufacturing systems*, 32(1):281–293, 2013.

[428] Hwai-En Tseng and Shih-Chen Lee. Disassembly sequence planning using interactive genetic algorithms. In *2018 14th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*, pages 77–84. IEEE, 2018.

[429] Yaping Ren, Chaoyong Zhang, Fu Zhao, Huajun Xiao, and Guangdong Tian. An asynchronous parallel disassembly planning based on genetic algorithm. *European Journal of Operational Research*, 269(2):647–660, 2018.

[430] Yongting Tian, Xiufen Zhang, Zehua Liu, Xingyue Jiang, and Junfang Xue. Product cooperative disassembly sequence and task planning based on genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 105(5):2103–2120, 2019.

[431] James C Chen, Yin-Yann Chen, Tzu-Li Chen, and Yu-Chia Yang. An adaptive genetic algorithm-based and and/or graph approach for the disassembly line balancing problem. *Engineering Optimization*, 54(9):1583–1599, 2022.

[432] WD Li, Kai Xia, Liang Gao, and K-M Chao. Selective disassembly planning for waste electrical and electronic equipment with case studies on liquid crystaldisplays. *Robotics and Computer-Integrated Manufacturing*, 29(4):248–260, 2013.

[433] Jun Guo, Jingcheng Zhong, Yibing Li, Baigang Du, and Shunsheng Guo. A hybrid artificial fish swam algorithm for disassembly sequence planning considering setup time. *Assembly Automation*, 39(1):140–153, 2018.

[434] Kai Xia, Liang Gao, Lihui Wang, Weidong Li, and Kuo-Ming Chao. A simplified teaching-learning-based optimization algorithm for disassembly sequence planning. In *2013 IEEE 10th International Conference on e-Business Engineering*, pages 393–398. IEEE, 2013.

[435] Kai Xia, Liang Gao, Lihui Wang, Weidong Li, Xinyu Li, and Winifred Ijomah. Service-oriented disassembly sequence planning for electrical and electronic equipment waste. *Electronic Commerce Research and Applications*, 20:59–68, 2016.

[436] Shana Smith, Greg Smith, and Wei-Han Chen. Disassembly sequence structure graphs: An optimal approach for multiple-target selective disassembly sequence planning. *Advanced engineering informatics*, 26(2):306–316, 2012.

[437] P Mitrouchev, CG Wang, LX Lu, and GQ Li. Selective disassembly sequence generation based on lowest level disassembly graph method. *The International Journal of Advanced Manufacturing Technology*, 80:141–159, 2015.

[438] Xiu Fen Zhang, Gang Yu, Zhi Yong Hu, Cheng Hui Pei, and Guo Qiang Ma. Parallel disassembly sequence planning for complex products based on fuzzy-rough sets. *The International Journal of Advanced Manufacturing Technology*, 72:231–239, 2014.

[439] Hsien-Pin Hsu. A fuzzy knowledge-based disassembly process planning system based on fuzzy attributed and timed predicate/transition net. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(8):1800–1813, 2016.

[440] E. Tuncel, A. Zeid, and S. Kamarthi. Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *Journal of Intelligent Manufacturing*, 25(4):647–659, 2014.

[441] Suleyman Mete and Faruk Serin. A reinforcement learning approach for disassembly line balancing problem. In *2021 International Conference on Information Technology (ICIT)*, pages 424–427. IEEE, 2021.

[442] Sangchul Woo and Yunsick Sung. Dynamic action space handling method for reinforcement learning models. *Journal of Information Processing Systems*, 16(5):1223–1230, 2020.

[443] Yuanjun Laili, Yulin Li, Yilin Fang, Duc Truong Pham, and Lin Zhang. Model review and algorithm comparison on multi-objective disassembly line balancing. *Journal of Manufacturing Systems*, 56:484–500, 2020.

[444] E.G. Kalaycilar, M. Azizoglu, and S. Yeralan. A disassembly line balancing problem with fixed number of workstations. *European Journal of Operational Research*, 249(2):592–604, 2016.

[445] Zsolt J Viharos and Richárd Jakab. Reinforcement learning for statistical process control in manufacturing. *Measurement*, 182:109616, 2021.

[446] Reinaldo AC Bianchi, Carlos HC Ribeiro, and Anna Helena Reali Costa. Heuristically accelerated reinforcement learning: Theoretical and experimental results. In *ECAI*, pages 169–174, 2012.

[447] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34:13550–13563, 2021.

[448] R.A.C. Bianchi, Jr. Celiberto, L.A., P.E. Santos, J.P. Matsuura, and R. Lopez De Mantaras. Transferring knowledge as heuristics in reinforcement learning: A case-based approach. *Artificial Intelligence*, 226:102–121, 2015.

[449] A.J.D. Lambert and S.M. Gupta. *Disassembly Modeling for Assembly, Maintenance, Reuse and Recycling.* CRC Press, Boca Raton, Florida, USA, 2004.

[450] Xiaoqun Liao, JiaYi Wang, and Li Ma. An algorithmic approach for finding the fuzzy constrained shortest paths in a fuzzy graph. *Complex & Intelligent Systems*, 7:17–27, 2021.

[451] Hu Qin, Xinxin Su, Teng Ren, and Zhixing Luo. A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management*, 8:370–389, 2021.

[452] Filipe Vital and Petros Ioannou. Scheduling and shortest path for trucks with working hours and parking availability constraints. *Transportation Research Part B: Methodological*, 148:1–37, 2021.

[453] Moritz Baum, Julian Dibbelt, Andreas Gemsa, and Dorothea Wagner. Towards route planning algorithms for electric vehicles with realistic constraints. *Computer Science-Research and Development*, 31:105–109, 2016.

[454] Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Science*, 53(6):1627–1655, 2019.

[455] Jonathan D Adler, Pitu B Mirchandani, Guoliang Xue, and Minjun Xia. The electric vehicle shortest-walk problem with battery exchanges. *Networks and Spatial Economics*, 16:155–173, 2016.

[456] Zeynel Abidin Çil, Hande Öztop, Zülal Diri Kenger, and Damla Kizilay. Integrating distributed disassembly line balancing and vehicle routing problem in supply chain: Integer programming, constraint programming, and heuristic algorithms. *International Journal of Production Economics*, 265:109014, 2023.

[457] Marlin W Ulmer, Justin C Goodson, Dirk C Mattfeld, and Barrett W Thomas. On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics*, 9(2):100008, 2020.

[458] Samah WG AbuSalim, Rosziati Ibrahim, Mohd Zainuri Saringat, Sapiee Jamel, and Jahari Abdul Wahab. Comparative analysis between dijkstra and bellman-ford algorithms in shortest path optimization. In *IOP Conference Series: Materials Science and Engineering*, volume 917, page 012077. IOP Publishing, 2020.

[459] Ismail H Toroslu. Improving the floyd-warshall all pairs shortest paths algorithm. *arXiv preprint arXiv:2109.01872*, 2021.

[460] R Dondo. A new formulation to the shortest path problem with time windows and capacity constraints. *Latin American applied research*, 42(3):257–265, 2012.

[461] Kairanbay Magzhan and Hajar Mat Jani. A review and evaluations of shortest path algorithms. *Int. J. Sci. Technol. Res*, 2(6):99–104, 2013.

[462] Florentin D Hildebrandt, Barrett W Thomas, and Marlin W Ulmer. Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Computers & operations research*, 150:106071, 2023.

[463] Weihang Dong, Wei Zhang, and Wei Yang. Node constraint routing algorithm based on reinforcement learning. In *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, pages 1752–1756. IEEE, 2016.

[464] Meng Qi, Mengxin Wang, and Zuo-Jun Shen. Smart feasibility pump: Reinforcement learning for (mixed) integer programming. *arXiv preprint arXiv:2102.09663*, 2021.

[465] Quentin Cappart, Thierry Moisan, Louis-Martin Rousseau, Isabeau Prémont-Schwarz, and Andre A Cire. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3677–3687, 2021.

[466] Wenwen Xia, Chong Di, Haonan Guo, and Shenghong Li. Reinforcement learning based stochastic shortest path finding in wireless sensor networks. *Ieee Access*, 7:157807–157817, 2019.

[467] Luuk Arts, Tom Heskes, and Arjen P de Vries. Comparing discretization methods for applying q-learning in continuous state-action space, 2017.

[468] Sean R Sinclair, Siddhartha Banerjee, and Christina Lee Yu. Adaptive discretization for episodic reinforcement learning in metric spaces. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–44, 2019.

[469] Jordan Baumgardner, Karen Acker, Oyinade Adefuye, Samuel Thomas Crowley, Will DeLoache, James O Dickson, Lane Heard, Andrew T Martens, Nickolaus Morton, Michelle Ritter, et al. Solving a hamiltonian path problem with a bacterial computer. *Journal of biological engineering*, 3:1–11, 2009.