algorithms

MDPI

Article

Intent-Based Resource Allocation in Edge and Cloud Computing Using Reinforcement Learning

Dimitrios Konidaris [1,*], Polyzois Soumplis [2], Andreas Varvarigos [3] and Panagiotis Kokkinos [1,2,*]

[1] Department of Digital Systems, University of Peloponnese, 23100 Sparta, Greece
[2] Institute of Communication and Computer Systems (ICCS), National Technical University of Athens, Zografou, 15772 Athens, Greece
[3] Department of Informatics and Telematics, Harokopio University of Athens, Kallithea, 17676 Athens, Greece
* Correspondence: dkonidaris@go.uop.gr (D.K.); p.kokkinos@go.uop.gr (P.K.)

Abstract

Managing resource use in cloud and edge environments is crucial for optimizing performance and efficiency. Traditionally, this process is performed with detailed knowledge of the available infrastructure while being application-specific. However, it is common that users cannot accurately specify their applications' low-level requirements, and they tend to overestimate them—a problem further intensified by their lack of detailed knowledge on the infrastructure's characteristics. In this context, resource orchestration mechanisms perform allocations based on the provided worst-case assumptions, with a direct impact on the performance of the whole infrastructure. In this work, we propose a resource orchestration mechanism based on intents, in which users provide their high-level workload requirements by specifying their intended preferences for how the workload should be managed, such as prioritizing high capacity, low cost, or other criteria. Building on this, the proposed mechanism dynamically assigns resources to applications through a Reinforcement Learning method leveraging the feedback from the users and infrastructure providers' monitoring system. We formulate the respective problem as a discrete-time, finite horizon Markov decision process. Initially, we solve the problem using a tabular Q-learning method. However, due to the large state space inherent in real-world scenarios, we also employ Deep Reinforcement Learning, utilizing a neural network for the Q-value approximation. The presented mechanism is capable of continuously adapting the manner in which resources are allocated based on feedback from users and infrastructure providers. A series of simulation experiments were conducted to demonstrate the applicability of the proposed methodologies in intent-based resource allocation, examining various aspects and characteristics and performing comparative analysis.

Keywords: resource allocation; intents; cloud computing; Q-learning; deep reinforcement learning; neural networks

check for updates

Academic Editor: Yousef Farhaoui

Received: 27 August 2025
Revised: 1 October 2025
Accepted: 1 October 2025
Published: 4 October 2025

Citation: Konidaris, D.; Soumplis, P.; Varvarigos, A.; Kokkinos, P. Intent-Based Resource Allocation in Edge and Cloud Computing Using Reinforcement Learning. *Algorithms* **2025**, *18*, 627. https://doi.org/10.3390/a18100627

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Cloud computing serves the computing workload and the associated data of services and applications from various domains, with diverse resource capacity, security, reliability, cost and other requirements. Lately, edge computing has gained increased interest from industry and academia, providing computation and storage capacity at the network periphery, close to where various devices and sensors produce data. Edge computing enables more rapid processing, decreasing the applications' experienced latency, while

also limiting the load that is carried and served by higher-layer resources in the cloud. Edge and cloud resources that operate in collaboration formulate what is known as the edge–cloud continuum [1], allowing the integration of the benefits of the different layers. In this continuum, the availability of resources increases as we move from the edge to the cloud, but so does the latency experienced by the applications. Therefore, tasks and data are allocated as follows: temporary storage and delay-sensitive computations are at the edge, while computational heavy processing and persistent storage are at the cloud.

Efficient resource allocation is a key factor for making the best use of infrastructures. Most existing approaches to resource allocation assume that workload requirements are known precisely, for instance, provided directly by users, and that orchestration mechanisms have complete knowledge of the properties and current state of the resources. In reality, however, these assumptions often do not hold. Users usually have a subjective understanding of their needs—for example, what one user considers low-cost or fast may differ for another application. Additionally, users often have only a vague view of the available infrastructure, either because resource providers do not supply complete information or because users find it burdensome to fully explore the details. Consequently, users cannot always express their requirements clearly in quantitative terms or align them accurately with the actual capabilities of the infrastructure. Similarly, orchestration mechanisms cannot always monitor resources in full detail due to their large number, heterogeneity, and constantly changing status. This issue is compounded when resources belong to different providers, as some may be unwilling to share detailed information about their systems. The challenge is particularly pronounced with numerous edge resources scattered across various locations, managed by multiple organizations or individuals, and exhibiting highly diverse characteristics.

Recently, intent-based operations have been considered by various stakeholders (providers, standardization organizations, and academia) [2,3] as a way for applications and users to express their requirements from digital infrastructures in terms of their computing, networking, storage and other requirements. In particular, it is important to make clear the difference between Quality of Services (QoS) and Quality of Experiences (QoE). QoS is a technical and objective measurement of how well a network or system delivers services like availability, latency, and speed. Conversely, QoE is a subjective measurement of the satisfaction or the frustration of the user of an application or service [4]. Additionally, the user's intent is the goal that the user wants to achieve without worrying about technical details [3].

Overall, the goal of the intent-based approach is to describe in an abstract manner an infrastructure's desired operation state, rather than the way of achieving it. In opposition, in Intent-Driven Networking, we focus in how a network can be regulated, automated and managed by the user's intent. An Intent-Driven Network refers to an intelligent networking paradigm capable of autonomously translating an operator's intents into actionable configurations, verifying and deploying them, while performing continuous optimizations to achieve the desired target network state [5]. Generally, in Intent-Driven Networking, we concentrate on how data will traffic more effectively in the network. Intent-Based Networking (IBN) seeks to simplify network management by automatically turning high-level, abstract service requests into precise policy definitions [6]. In other words, traffic optimization and safety is the main goal in Intent-Driven Networking, while in Intent-Driven Cloud/Edge Computing, the most important goal is the optimization of application execution performance and cost. In other words, in cloud/edge computing, we are interesting in how the application will operate in order to satisfy the user's intent.

In this work, we focus on intent-based resource allocation for edge and cloud computing infrastructures (Figure 1). The key idea is that application requirements are expressed

as intents in a way that is independent of the underlying infrastructure, since application owners are often unable to specify precise quantitative characteristics or numerical requirements for their workloads. The main contribution of our approach is the development of an advanced resource allocation mechanism that integrates Markov Decision Processes (MDPs), tabular Q-learning and Deep Reinforcement Learning to translate the users'/applications' intentions to efficient resource allocations within the edge–cloud continuum. In the discrete-time, finite-horizon MDP model considered, each state corresponds to a different set of past decisions and each action to a specific resource allocation decision. The tabular Q-learning approach, while allowing us to calculate optimal Q-values, becomes inefficient as the state space grows. To address this limitation, we employ neural networks in a Deep Reinforcement Learning approach. This enables a continuous improvement in the performed allocations by incorporating user satisfaction and real-time feedback from infrastructure monitoring, both of which are depicted in the reward function of our RL approach. The simulation results obtained showcase the validity of our approach.

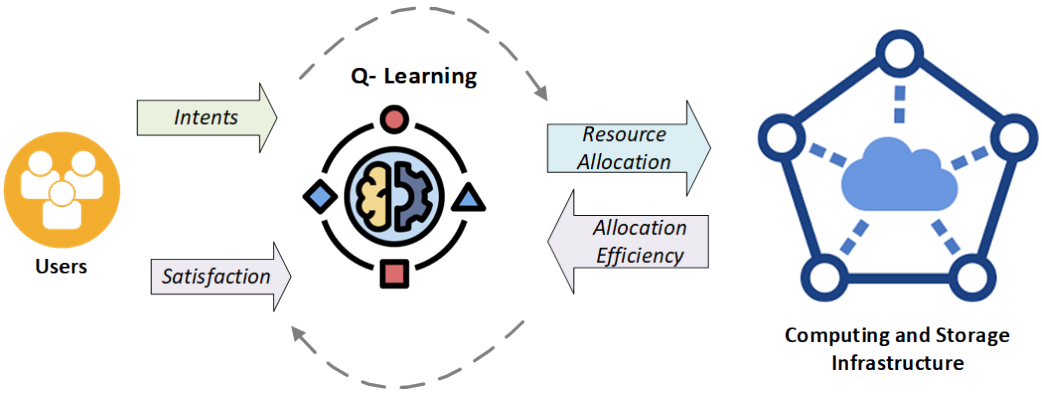


Figure 1. Intent-driven resource allocation in edge and cloud computing.

Scope and limitations. This study is intended as a proof of concept for intent-driven resource orchestration. To keep the evaluation tractable, we focused on a single-user setting, relatively modest resource scales (up to 25 resources), and a simulated Gym-based environment. We also modeled resources in a simplified, binary form to highlight the feasibility of mapping intents to allocations. Extending the framework to multi-user scenarios, multi-dimensional heterogeneous resources, and experiments with public cluster traces represents an important direction for future work.

The reminder of this paper is organized as follows. Previous work is reported in Section 2. In Section 3, we discuss the system model and the infrastructure-agnostic operations. In Section 4, we describe our Reinforcement Learning methodologies for intent-driven resource allocation. Simulation experiments are presented in Section 5. Finally, we conclude our work in Section 6.

2. Related Work

Resource management is widely recognized as a critical aspect of edge and cloud computing, and numerous approaches have been proposed to address it [7]. In recent years, Reinforcement Learning (RL) methods have gained increasing attention as a means to perform efficient resource allocation in these environments.

Reinforcement Learning (RL) is a branch of machine learning that has attracted considerable attention in recent research. In RL, one or more agents engage with the environment to understand its behavior [8]. The agent selects actions and obtains feedback through rewards, aiming to maximize the cumulative reward over time. To achieve this, the agent evaluates both immediate and future potential rewards to determine the optimal policy.

This involves an iterative process where the agent continuously updates its actions based on received rewards: if the cumulative reward is positive, the agent maintains its current policy, whereas negative rewards prompt the agent to adjust its strategy to improve outcomes in subsequent iterations.

RL methodologies can be categorized as model-based or model-free. In model-free RL, agents learn to make decisions without an explicit model of the environment, relying on trial and error. Common model-free methods include Q-learning, SARSA (State–Action–Reward–State–Action), Monte Carlo methods, TD learning (Temporal Difference learning), Actor–Critic methods, and Deep RL. These approaches are particularly useful when modeling the environment is difficult, though they often require more training data, computational resources, and energy to learn optimal policies compared to model-based methods. Deep RL, which integrates RL with neural networks, excels in handling high-dimensional state spaces and is well suited for complex scenarios where traditional RL struggles. In [9], a Q-learning-based RL approach is proposed to translate user intentions into efficient resource allocations within a cloud infrastructure, continuously improving resource distribution based on both user satisfaction and infrastructure efficiency.

RL has been applied to a wide range of problems, including zero-sum games [10], stock market forecasting [11], and decision-making in autonomous driving [12]. RL-based methods have also been used for resource allocation in wireless networks, such as avoiding interference from hidden nodes in CSMA/CA [13], 5G service optimization using deep Q-learning [14–16], hybrid networks combining RF and visible light communications [17], satellite–terrestrial networks [18], and optical networks [19]. In the edge–cloud domain, RL and Deep RL (DRL) techniques have been widely explored [15,20–25]. For example, ref. [15] proposes an integrated approach for assigning tasks and allocating resources within a multi-user WiFi-enabled mobile edge computing environment, while [21] introduces a Q-learning method for efficient edge–cloud resource allocation in IoT applications. In [22], neural networks are used for computational offloading in edge–cloud systems. Ref. [23] applies DRL to balance mobile device workloads in edge computing, reducing service time and task failures. Ref. [24] uses a model-free DRL approach to orchestrate edge resources while minimizing operational costs, and [25] proposes an RL-based task scheduling algorithm for load balancing and for reducing the energy consumed. Despite these advances, deploying Deep Q-learning Networks (DQNs) presents challenges, such as the reliance on neural networks for Q-value approximation, which can be sensitive to data quality and quantity [26], and the critical selection of hyperparameters during network initialization to ensure faster convergence and efficient learning [27].

Intent-driven operations aim to simplify the management of complex infrastructures by separating the "what" of user goals from the "how" of resource orchestration. Originally applied to networks under the term intent-based networking [5,28–30], intents are often described using structured formats like JSON or YAML, though Large Language Models (LLMs) have also been explored for this purpose [31].

In particular, the *IntentContinuum* framework [32] demonstrates how LLMs can parse free-text intents and directly interact with resource managers to maintain service-level objectives across the compute continuum. This highlights the potential of LLMs for richer and more flexible intent parsing, complementing our Reinforcement Learning methodology that focuses on intent-to-resource translation and allocation.

The use of intents is now being extended to cloud and edge computing [33–38]. For example, ref. [33] defines rules for expressing service-layer requirements, while [34] provides a Label Management Service for modeling policy requirements. In [35], an intent-aware, learning-based framework for the offloading of tasks is developed for air–ground integrated vehicular edge computing. Ref. [36] presents a framework translating cloud performance

intents into concrete resource requirements, and [37] introduces a Service Intent-aware Task Scheduling (SIaTS) framework that uses auction-based scheduling to match task intents with computing resources. Ref. [38] proposes an intent-driven orchestration paradigm to manage applications through service-level objectives, ref. [39] matches multi-attribute tasks to cloud resources, and [6] automates virtual network function deployment over cloud infrastructure using intent-based networking.

A recent survey by [40] provides a systematic review of Deep Reinforcement Learning approaches for resource scheduling in multi-access edge computing (MEC). The study categorizes existing work according to the system model, algorithmic family, and evaluation methodology, highlighting edge-specific constraints such as device mobility, limited energy, and heterogeneous latency requirements. It also compares common metrics and benchmarks used across the literature, which helps position our proof-of-concept evaluation based on synthetic tasks relative to the broader ecosystem of MEC scheduling research. In contrast, our formulation focuses on intent-driven resource orchestration and explicitly incorporates user satisfaction into the reward design, thereby addressing a different but complementary aspect of edge–cloud resource management.

DRL Surveys and Positioning

Recent comprehensive surveys synthesize the growing literature on Deep Reinforcement Learning (DRL) for cloud/resource scheduling and highlight common design patterns, benchmarks, and open challenges. In particular, ref. [41] provides a structured review of DRL approaches for cloud scheduling, discusses multi-dimensional resource models, and compares Q-learning/DQN families against actor–critic, model-based and multi-agent methods. This survey helps position our work within the DRL for the scheduling landscape and motivates the need to consider alternative algorithm classes (policy-gradient, actor–critic, and MARL) in future extensions.

Our work builds on these ideas but takes a distinct approach by applying Q-learning-based methods to translate high-level user intentions into low-level resource allocations in an edge–cloud infrastructure. We compute optimal Q-values and also evaluate a neural network-based method designed to improve scalability for more realistic scenarios, highlighting differences in efficiency and performance.

3. System Model and Infrastructure-Agnostic Operations

3.1. Infrastructure

Our work is based on a computing environment, consisting of multiple resources (N) located both at the edge and the cloud. Each resource has different capabilities regarding capacity, usage cost, and security. Capacity refers to the computing power or the storage capabilities of the unit, measured e.g., as the number of (virtual) CPUs a computing resource has or the number of Gigabytes (GB) available in a storage resource. The cost of use can be defined in several ways, compatible with the cloud computing paradigm, such as a fixed fee for a period of time or charging based on the quantity used and time utilized (a pay-as-you-go (PAYG) model), e.g., GBs per hour. Security depends on the specific mechanisms provided by each resource, for example, access control policies, encryption capabilities, or authentication protocols. Additional parameters of interest may also be taken into account.

These parameters are assumed to take discrete values chosen from predefined sets, which aligns with the existing cloud computing services. In practice, public clouds provide a range of virtual computing instance types that combine computing (virtual CPU), memory, storage, and networking resources in different capacities and are optimized for distinct workload profiles, such as computing and memory- or storage-intensive tasks [42]. In

this setting, the virtualized infrastructure resources are modeled with capacity, cost, and security levels, each taking values from these discrete sets (namely "resource levels"):

- N_c levels of resource capacity: $S_C = \{c_1, c_2, \ldots, c_{N_c}\}$;
- N_u levels of resource cost: $S_U = \{u_1, u_2, \ldots, u_{N_u}\}$;
- N_e levels of resource security: $S_E = \{e_1, e_2, \ldots, e_{N_e}\}$.

3.2. User Workloads

Workload requests are defined by users or applications that require computational power or storage and are submitted to an orchestration entity responsible for managing the infrastructure.

Workload requests are generated by users and applications in a continuous manner requesting computing and storage resources for their service. These requests are submitted in an orchestration system (e.g., kubernetes) that manages the infrastructure. Each such request contains the requirements for the proper execution of the task under submission. These requirements include the required computing capacity (e.g., the number of virtual CPUs) for its proper execution or the size of the data that need to be stored (e.g., 2 GB). In addition, it can capture requirements that are independent of the underlying infrastructure, for instance, constraints or preferences related to cost (e.g., based on the overall available budget), security (e.g., based on compliance requirements), and performance (e.g., in the case of a mission-critical application), in the form of intents. This may be necessary when the user or application cannot express these aspects quantitatively. These intents may be expressed in different ways, e.g., by characterizing the desire for "fast" execution, "highly secure" storage, or "low-cost" operation. In our work, we capture this abstraction by defining a small set of "intent levels" for each operational dimension (e.g., cost, security, etc.):

- $\hat{N}_c$ levels of user intent capacity: $\hat{S}_C = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{\hat{N}_c}\}$, where $\hat{N}_c << N_c$;
- $\hat{N}_u$ levels of user intent cost: $\hat{S}_U = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_{\hat{N}_u}\}$, where $\hat{N}_u << N_u$;
- $\hat{N}_e$ levels of user intent security: $\hat{S}_E = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_{\hat{N}_e}\}$, where $\hat{N}_e << N_e$.

We assume that these "intent levels" of a user/application are much smaller in number than the "resource levels" (Section 3.1), based on the reasonable idea that a user has a very abstract view of the resource's characteristics and their granularity (Figure 2). An intent level (IL) can be loosely viewed as a Class of Service (CoS) level, with the important difference that it is a subjective measure of desired performance compared to CoS, which is objective. A CoS usually corresponds to some specific quantitative level of performance, while an IL captures the way the user/application perceives this performance, which is a subjective criterion (a given objective CoS may be interpreted as different ILs by different users). One of the targets of our approach is to learn the (objective) CoS that corresponds to the IL asked by a given user or application and allocate the resources required to implement the respective (objective) CoS for the user.

The matching of these intent levels that are infrastructure-agnostic to the various resource levels (Section 3.1) is critical for intent-based operations. This is the research challenge in our work. So, the j-th submitted workload of user k, w_{jk}, can be described with the intent tuple $\hat{X}_{jk} = \{\hat{c}_{jk}, \hat{u}_{jk}, \hat{e}_{jk}\}$, where $\hat{c}_{jk} \in \hat{S}_C$, $\hat{u}_{jk} \in \hat{S}_U$ and $\hat{e}_{jk} \in \hat{S}_E$. Then, the user intents are matched to the appropriate resource levels in the infrastructure in a matching process that can be described through transition function f. Particularly, function f takes as input the user-specified intent parameters ($\hat{c}_{jk}$, $\hat{u}_{jk}$, and $\hat{e}_{jk}$), corresponding to the desired capacity, cost, and security levels for the workload w_{jk} and maps them to the most suitable infrastructure-related resource levels c_i, u_i and e_i from the available resource pool. This

multi-dimensional process for a particular user or application a can be described through function f_a, which is defined as follows:

$$f_a : (\hat{c}_{jk}, \hat{u}_{jk}, \hat{e}_{jk}) \rightarrow (c_i, u_i, e_i)$$

However, aligning the user's intent with the available resource characteristics can be a difficult process. For instance, a request for high computational capacity ($\hat{c}_{jk} = \text{High}$) at a low cost ($\hat{u}_{jk} = \text{Low}$) can be a challenge. Function f is not pre-defined but instead is dynamically calculated for each user/application through the proposed mechanism.

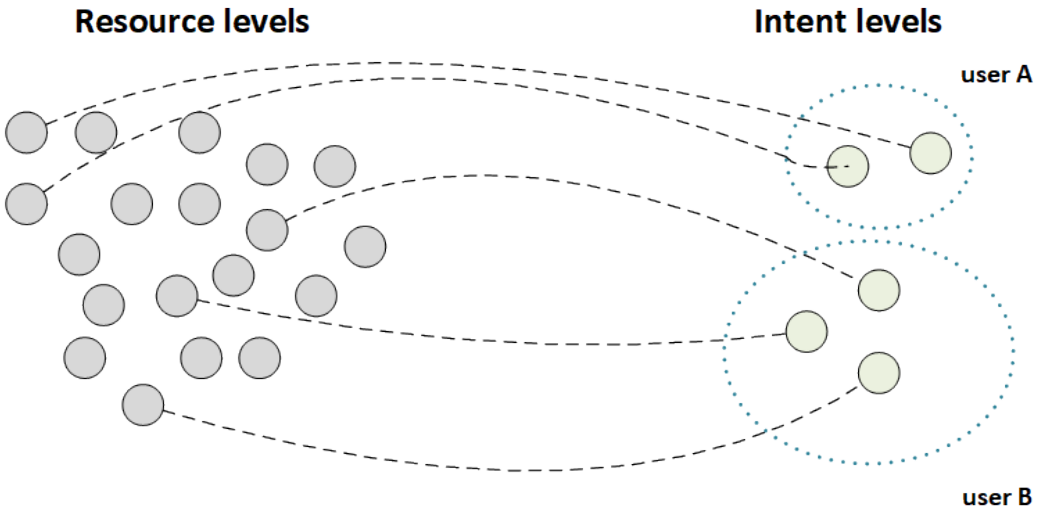


Figure 2. An example of resource and intent levels for a parameter of interest.

3.3. Example of Infrastructure-Agnostic Operation

In what follows, we present an example of an infrastructure-agnostic operation that involves a data storage request, which the infrastructure should serve. The request parameters include the data storage requirements, which are expressed in Gigabytes (GB). In addition, the request's parameters include an associated intent tuple, R, indicating that the data should be stored in a resource, which has "low" cost and increased ("high") security characteristics: $R = \{\hat{u}, \hat{e}\} = \{"low", "high"\}$. We can consider a scenario where the cost and security parameters' "intent levels" are $\hat{S}_U = 1, 2, 3$ and $\hat{S}_E = 1, 2, 3$, respectively, and the "low"-cost intent corresponds to value 1 of $\hat{S}_U$, while the "high"-security intent matches value 3 of $\hat{S}_E$. For simplicity, we can express these intents with a tuple, $R = \{\hat{u}, \hat{e}\} = 1, 3$.

The objective of the procedure presented is to effectively match the given intents into concrete resource allocation decisions regarding the service of the tasks/workload. The aim of the approach outlined below is to convert in an efficient manner the specified user intents into concrete actions for allocating resources, executing tasks or storing data. Efficiency has to do with aligning, as closely as possible, the resource-related actions (e.g., allocate resources) in the infrastructure with the actual intentions of the users or applications, regarding the service their tasks or data are receiving from it. For instance, we consider the scenario of an infrastructure provider offering resources with $N_u = 8$ cost ("resource") levels for data storage service with different capabilities, expressed in terms of monthly cost per GB, e.g., $S_u = 5, 15, 25, 35, 45, 55, 65, 75$. For a user that specifies its intention for the service's cost as $\hat{u} = 1$ (e.g., indicating that it is looking for a relatively low-cost service), the methodology needs to map this intention and the associated "intent level" to an actual cost from the ones available from the set S_u. In general, we might expect that $\hat{u} = 1$ corresponds to 5, 15, or even 25 monthly costs per GB ("resource level"). In reality, however, this "intent level" varies from user to user and may align with one of the available "resource

levels," or in some cases, it might not match any existing level. Thus, the same intent level from different users may map to different resource levels (Figure 3). This is due to the fact that different users may hold varying perceptions of what constitutes, for example, a "low"-cost resource.

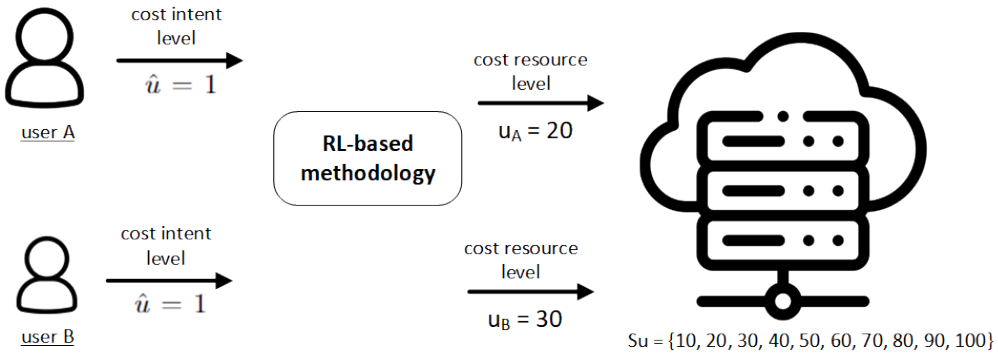


Figure 3. Example of intent-driven resource allocation.

4. Q-Learning-Based Intent Translation

We utilize Reinforcement Learning (RL) to automate the process of identifying a policy that matches users' intent levels with the appropriate resource levels for specific parameter types. We formulate the problem as a discrete-time Markov Decision Process (MDP). This MDP is characterized by the tuple (S, A, P, R, γ), where (i) S is a finite set of states representing the various distributions of infrastructure resources (utilization), (ii) A denotes the set of allowable low-level actions (allocations/deallocations of resources) that facilitate state transitions during the application's resource allocation, (iii) P is the state transition probability matrix, indicating the probability of transitioning from one state to another following a specific action, (iv) R is the reward function that assigns an immediate numerical reward to each state–action pair (s, a, s'), and (v) γ is the discount factor that harmonizes the emphasis on immediate rewards with future rewards in long-term reward optimization.

4.1. State, Action Space and Rewards

The state space S, action space A, and reward r are the fundamental components that need to be defined in our RL-based method. The RL process is executed at time $t \in T$, where T is a set of discrete time steps.

The **system state** $s \in S$ at time t represents the availability of resources within the cloud infrastructure. For clarity, we consider that each task or workload exclusively occupies one of the N available resources. Consequently, the environment is modeled as a tuple capturing the availability of resources:

$$s = \{o_1, \ldots, o_N\},$$

where each $o_i \in \{-1, 1\}$ denotes whether resource i is occupied or it is free. For more clarity of exposition, we adopt a binary abstraction of the infrastructure, where each resource is modeled as either occupied ($o_i = 1$) or free ($o_i = -1$). This simplification allows us to highlight the feasibility of intent-to-resource mapping while keeping the state space tractable. In practice, resources exhibit multiple continuous attributes (e.g., CPU, memory, storage, bandwidth) that can be incorporated in extended formulations.

4.1.1. Extension to Continuous and Multi-Dimensional Resources

The binary abstraction used in this work, where each resource is modeled as either free or occupied, was chosen to keep the proof of concept tractable. However, real infrastructures are inherently multi-dimensional: each resource is characterized by multiple attributes such as CPU cores, memory, storage capacity, and I/O throughput. A more realistic representation can be achieved by extending the state vector to continuous or multi-level dimensions.

Formally, the state of resource i can be expressed as:

$$o_i = \left(c_i, m_i, s_i, b_i\right),$$

where $c_i \in \mathbb{R}^+$ denotes available CPU capacity, $m_i \in \mathbb{R}^+$ the available memory, $s_i \in \mathbb{R}^+$ the storage capacity, and $b_i \in \mathbb{R}^+$ the bandwidth. The global system state then becomes a concatenation of such vectors across all resources. Actions similarly generalize to multi-resource allocations or migrations of tasks requiring bundles of heterogeneous resources.

While this extended model is more expressive, it also enlarges the state–action space exponentially, which motivates the adoption of scalable Deep Reinforcement Learning methods or hierarchical/policy-gradient approaches. Exploring such extensions represents a promising direction for future work, beyond the binary abstraction adopted in this study.

The **action space**, A, encompasses all actions that may be executed from the various states and establishes the rules governing state transitions. As the agent interacts with the environment, it tests different actions in order to identify those that most effectively fulfill the defined intent-based objectives. In our study, we consider that at any given time t, two types of actions can be performed: either assigning a new user task to an available resource or migrating an ongoing task to another free resource. Accordingly, the set of possible actions can be expressed as $A_t = \{r_1, \ldots, r_N\}$, where N denotes the total number of available resources. However, not every transition between states is feasible in practice, since we limit each time step to handle a single new task assignment or a single task migration to another resource. For instance, in a cloud environment with $N = 4$ computing units (resources), consider state $s_1 = \{1, 1, -1, -1\}$, where the first and second resources serve a task.

In this case, a valid action could lead to state $s_2 = \{1, -1, 1, -1\}$, representing a migration of the task from the second to the third resource. On the other hand, state $s_3 = \{-1, -1, 1, 1\}$ cannot be realized, since it requires more than one task to migrate to different resources in the same time step.

When the agent executes an action, a, from state s at time t, leading to a transition to state s', it receives a **reward** $R(s, a, s')$, which quantifies the effect of the action taken. A key innovation in our approach is that the reward is influenced not only by the state of the infrastructure (as is common in related works) but also by feedback from the user submitting the task. On the user side, the reward reflects how well the executed task aligns with the user's intent, capturing their level of satisfaction. On the infrastructure side, it measures how efficiently the available resources are utilized. These two aspects are interconnected since the inefficient use of resources can lead to uncompleted tasks, which in turn decreases user satisfaction. In practice, user satisfaction can be captured through immediate feedback mechanisms, such as a user interface [2], once a task has been processed, while the efficiency of the infrastructure can be tracked via monitoring systems, e.g., Prometheus.

In our work, we consider the following reward function, $R(s_t, a_t, s_{t+1}) = a \cdot \varphi(a_t) + b \cdot ul(s_{t+1})$, where φ is the satisfaction level based on the action performed at time t and $ul(s_{t+1})$ the utilization of the resources at the subsequent state s_{t+1}. Specifically, φ is defined as the discrepancy between the intended parameters $\hat{C}_{jk}$, cost $\hat{U}_{jk}$, and security $\hat{E}_{jk}$ for a

specific workload w_{jk} submitted by user k and the actual resource levels $C_{i,j,k}$, $U_{i,j,k}$, and $E_{i,j,k}$ allocated by infrastructure i. The satisfaction function φ can be defined as follows:

$$\varphi(\hat{X}_{jk}, X_{i,j,k}) = \frac{1}{1 + \sum \left| \hat{X}_{jk} - X_{i,j,k} \right|}$$

Coefficients a and b define the balance between resource utilization and user satisfaction feedback. The satisfaction score is computed quantitatively by comparing the user's intended preferences with the characteristics of the resources to which the tasks are assigned, such as cost or security levels. In real-world scenarios, however, the satisfaction reported by a user is inherently subjective and may differ from this calculated "ideal" value.

4.1.2. Noisy and Delayed Feedback

In the reward function presented above, user satisfaction is modeled through a quantitative comparison of intended versus allocated resource levels. This formulation assumes that feedback is immediate and noise-free. In realistic deployments, however, user feedback may be delayed or inherently noisy, due to a subjective evaluation or measurement uncertainty.

To capture this aspect, the satisfaction term can be perturbed with a stochastic noise component $\epsilon \sim \mathcal{N}(0, \sigma^2)$ so that

$$\tilde{\varphi}(\hat{X}_{jk}, X_{i,j,k}) = \varphi(\hat{X}_{jk}, X_{i,j,k}) + \epsilon.$$

Delayed feedback can be represented by applying the reward update not at time step t, but at $t + \delta$, with δ denoting the feedback delay. Preliminary experiments with Gaussian noise injection showed that the Deep RL agent maintained convergence trends, albeit with slower stabilization compared to the noise-free case. These robustness considerations indicate that while our abstraction provides a tractable proof of concept, extending the evaluation to incorporate noisy and delayed feedback is critical for real-world intent-based orchestration scenarios.

4.2. Q-Learning Methodology

Q-learning is a trial-and-error-based AI technique where a software agent learns to make optimal decisions. It does this by repeatedly interacting with its environment—which includes the system infrastructure and the user—to infer the user's intentions and learn a predictive value for every possible action. This "action–value function" allows the agent to identify which actions will best serve the user's goals and yield the greatest long-term rewards, all without requiring a pre-defined model of how the environment works.

For a state–action pair (s_t, a_t) at time t the optimal action–value function $Q^*(s_t, a_t)$ represents the expected cumulative reward when starting from state s_t, taking action a_t, and subsequently following the optimal policy. Similarly, the optimal value function, $J^*(s_t)$, for state s_t at time t provides the expected return when starting from state s_t and following the optimal policy. These functions are related by the following equation for each time step t within horizon T:

$$J^*(s_t) = \max_{a_t \in A} Q^*(s_t, a_t)$$

The Q-learning algorithm iteratively updates the estimate of the action–value function for each state–action pair visited by the agent, guided by the Bellman equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right],$$

where $Q(s_t, a_t)$ represents the estimated value of taking action a_t in state s_t, α is the learning rate that determines the weight assigned to newly acquired information, $R(s_t, a_t, s_{t+1})$ denotes the immediate reward obtained from the environment for executing action a_t, and γ is the discount factor that balances the importance of future rewards relative to immediate ones. In the tabular approach, these Q-values for all state–action pairs are stored in a data structure known as the Q-Table.

Various strategies can be employed by the agent to select actions in each state. These include random selection, choosing the least frequently executed action, or selecting the action associated with the highest Q-value. Additionally, many Q-learning implementations incorporate a probability parameter, ϵ, which governs the exploration–exploitation trade-off: it controls whether the agent chooses the action with the highest estimated value (exploitation) or explores alternative actions (exploration). The rewards obtained from these actions lead to continuous updates of the Q-values in the Q-Table, allowing the agent to refine its policy over time.

The cumulative reward at the different time steps t is defined as

$$G_t = R(s_t, a_t, s_{t+1}) + \gamma R(s_{t+1}, a_{t+1}, s_{t+2}) + \gamma^2 R(s_{t+2}, a_{t+2}, s_{t+3}) + \cdots,$$

where $R(s_t, a_t, s_{t+1})$ represents the immediate reward obtained by the agent at time step $t+1$ for executing action a_t from state s_t, while γ denotes the discount factor. The agent's objective is to determine the policy that maximizes the expected total reward G_t across all potential sequences of states and actions.

4.3. Neural Network-Based Approximation of Q-Values

The use of a Q-Table for storing state–action values can become demanding, due to the exponential growth of the table's size, when considering infrastructures/environments with a large number of resources and tasks. This is particularly true in an edge computing environment involving numerous small capacity resources and under the cloud-native (microservices) computing paradigm where the various applications consist of several subtasks.

A common method to perform a Q-value approximation involves training a neural network, leveraging its ability to generalize from visited to unvisited states and enabling the agent to learn complex mappings from states to Q-values, thereby estimating unknown Q values ($\hat{Q}(s, a)$) based on the Bellman optimality principle.

In this case, the neural network is trained with the collected data that are used to depict experiences, structured as tuples of the form (s, a, r, s'). This includes an iterative process with a varying number of steps, whose number depends on when it converges to the optimal Q-values $Q^*(s, a)$. In this way, the neural network learns from interactions with the environment and updates the $\hat{Q}$ values to reflect learned experiences at each iteration.

$$Q_{\text{target}}(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}),$$

When the aforementioned process is completed, the neural network is able to provide the approximated Q-values, denoted as $\hat{Q}(s, a)$, which are an estimation for the optimal action–value function. Leveraging this Q-values, the current state, s_t, at any given time t can be evaluated in conjunction with all possible actions $a \in A$ so as to calculate the estimated value function $\hat{J}(s_t)$ of the current state.

$$\hat{J}(s_t) = \max_{a \in A} \hat{Q}(s_t, a)$$

5. Performance Evaluation

In our experiments, we used Gym open source Python 3.12 (64-bit) library to create a custom environment that represents an edge–cloud infrastructure where tasks can be assigned and migrate to and from resources.

In particular, we consider a cloud infrastructure that contains storage resources, with $N_c = 10$ capacity resource levels, $S_c = \{10, 20, \ldots, 100\}$ GB, and with different combinations of the available characteristics, with respect to cost and security levels. We also assume that users specify the required storage capacity in a quantitative manner, while the other parameters (cost and security) are qualitatively specified using intent values. We performed a large number of experiments, employing different scenarios for the translation of intents to resource levels, matching various user intentions. One thing that stands out is that there is not necessarily a linear relation between the resource levels and the users' intents. This means for example that an intent level $\hat{TU}_1$ is not necessarily equal to TU_1, but depends on the user intention or notion of what fast, small, low, etc., means. In what follows, we consider a single user who issues storage task requests using intents, without knowledge on the underlying infrastructure.

In the experiments performed initially, we evaluated the Q-Table-based Q-learning methodology. We assumed resources of different cost resource levels, (i) low resource granularity (three levels), (ii) moderate resource granularity (five levels), and high resource granularity (seven levels), while the task requests generated were associated with two intent levels. The training procedure was executed for more than 10,000 timesteps. The parameters of Bellman's Equation were set to $\alpha = 0.5$, $\gamma = 0.9$, and $\epsilon = 0.5$. Figure 4 shows the evolution of the average reward during the first 1000 timesteps. In every scenario, the reward grows within the first 100 timesteps and then reaches a steady state, with only marginal improvements observed until timestep 10,000 (not shown in the figure). This behavior is expected, given the learning dynamics and the choice of $\epsilon = 0.5$, which implies that roughly half of the actions are chosen randomly rather than according to the computed Q-values. A key observation is the influence of the number of cost resource levels on the learning process: when the number of resource levels increases, the average reward decreases; conversely, fewer levels lead to higher rewards. The reason is that with fewer resource levels, the correspondence between intent and resource levels is easier to identify.

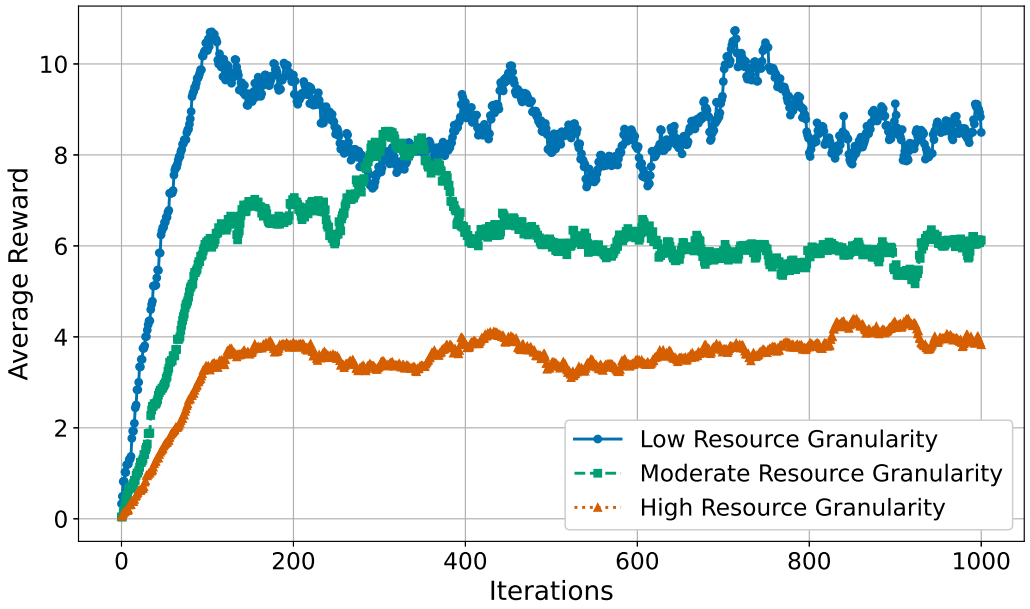


Figure 4. The average reward over time for different cost resource levels.

Figure 5 illustrates the average reward over time for different values of $\epsilon \in \{0.1, 0.2, \ldots, 0.9\}$ over 10 k timesteps. The ϵ parameter determines how fast the agent explores the environment and determines the optimal policy. The figure illustrates that for $\epsilon = 0.2$, the reward takes the highest value, indicating that this is the optimal value for the particular problem and the goals that have been set. Another possible strategy is to dynamically change the value of ϵ, where the strategy can be to initially select a high ϵ value to allow more exploration and then gradually decrease it in order to exploit the calculated Q-values.

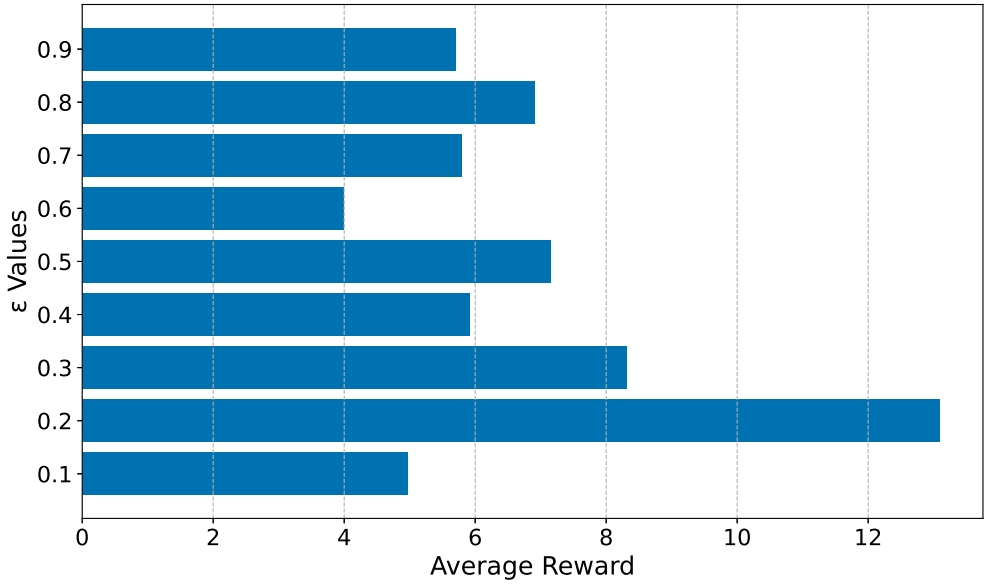


Figure 5. The averaged reward over time for varying values of exploration rate ϵ.

Another important aspect we investigated was the results that multiple intent parameters (e.g., cost, security) showed in the training process (Figure 6). An increased number of different intents that a user provides in a single task request results in a smaller average reward and a more gradual increase in its value over time. This complexity stems from the increased difficulty in correlating an expanding set of intents with the actual, infrastructure-aware values of the respective resources' parameters.

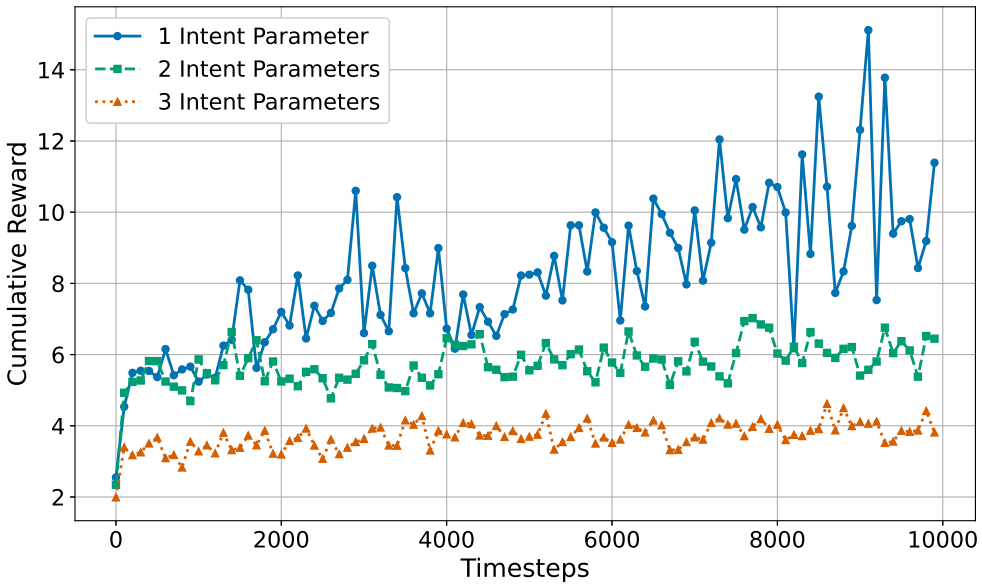


Figure 6. The average reward over time for a different number of intent parameters.

Next, we considered using neural networks in combination with Q-learning (Deep Reinforcement Learning) for value approximation in order to access how efficiently the Q-learning methodology interprets a user's intentions for its submitted infrastructure-agnostic requests to specific infrastructure-related decisions. Our experiments focused on the training phase of the Q-learning mechanism, comparing the Q-learning-based Reinforcement Learning (RL).

(Q-Table-based Q-learning) and Deep RL-DRL (Q-learning combined with a neural network) methodologies. In our experiments, reported in Figure 7, we initially assume an infrastructure with 25 resources, where both methodologies run for 500 episodes. Each episode comprises a sequence of states, actions, and rewards and concludes upon reaching a terminal state. As the figure illustrates, in the first 20 episodes, the two methodologies exhibit almost the same average cumulative rewards. After episode 100, the DRL methodology is more effective, yielding better rewards that are even five times higher. These results illustrate the effectiveness of using neural networks with Q-learning in order to identify a user's intent. This is particularly important when the state and action space sizes are large, which is the case when we have many resources, resource and intent levels and tasks. Also, the Q-learning mechanism is computationally expensive since it requires a lot of memory to store the Q-values and a lot of time to fill the Q-Table in comparison to the deep Q-learning methodology that uses a neural network.

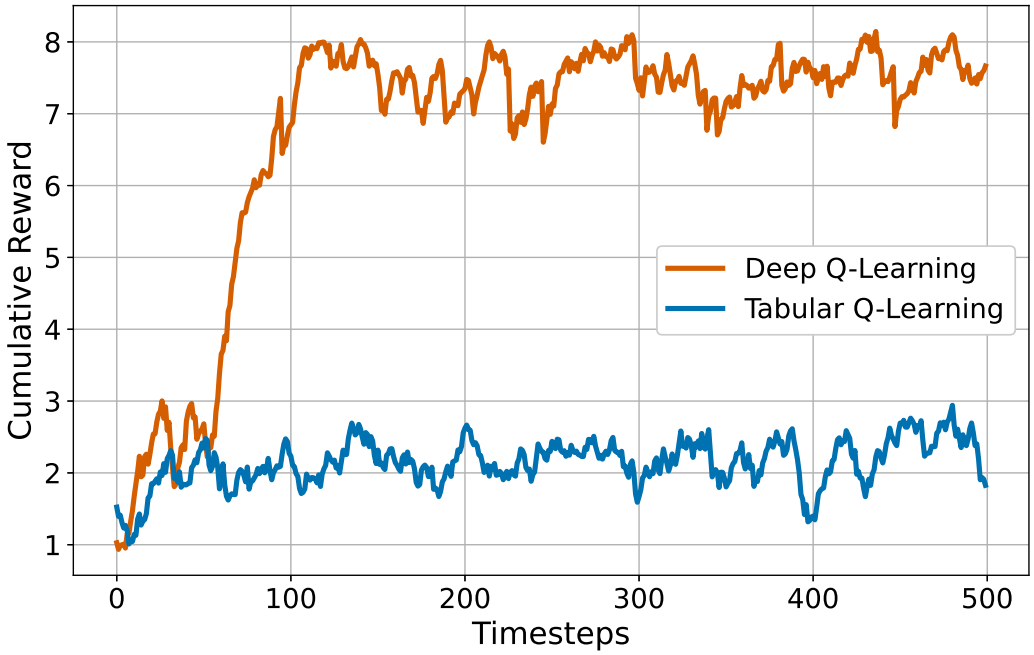


Figure 7. Q-Table-based Q-learning (RL) vs. Q-learning with a neural network (DRL) methodologies for 25 resources.

Also, we evaluated the reward achieved for the RL and DRL methodologies with a different number of resources (Figure 8). In particular, we considered 10, 13, 15, 19 and 25 resources and focused specifically on episode 200, where, in all cases, we observed that the achieved reward stabilizes. From the figure, we observe that for a small number of resources, the rewards achieved are comparable, while for a higher number of resources, the DRL methodology clearly outperforms the RL methodology, indicating its ability to more efficiently identify a user's intentions.

We can see the results in Table 1, where it is very clear that the reward in Deep RL is better than Q-learning for 19 and 25 resources.

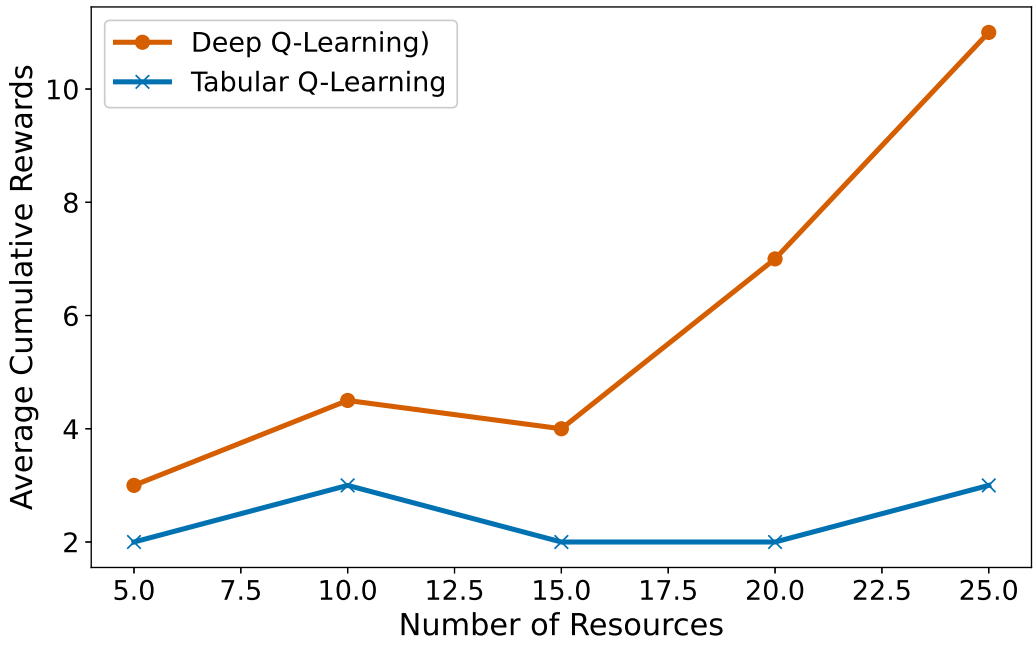


Figure 8. Q-Table-based Q-learning (RL) vs Q-learning with a neural network (DRL) methodologies for a different number of resources in episode 200.

Table 1. Tabular Q-learning vs Deep (Neural) Network.

Resources	Q-Learning Reward	Neural Network Reward
10	3	1.5
13	1.5	3.3
15	1.4	3.3
19	2	6
25	2	11

5.1. Baselines and Alternative RL Methods

In addition to comparing tabular Q-learning with a Deep RL variant, it is natural to consider baseline heuristics and other RL algorithms. A simple greedy allocation strategy, which always selects the resource that minimizes the immediate intent–resource mismatch, can serve as a non-learning baseline. Policy-gradient methods such as PPO or DDPG, as well as multi-agent or hierarchical RL, have been successfully applied in the recent resource scheduling literature.

Due to hardware and time constraints, we did not include full-scale implementations of these methods in this study. Nevertheless, we acknowledge their potential advantages in terms of sample efficiency and scalability, and we plan to incorporate them as additional baselines in future work.

The experiments were implemented in a custom Gym-based environment. Table 2 summarizes the key hyperparameters and the neural network architecture used in the DRL approach. In all scenarios, we trained the agents for up to 500 episodes, with the largest infrastructure configuration including 25 resources. The neural network employed two hidden layers with 64 neurons each and ReLU activation functions, while the output layer was linear. For optimization, we used the Adam algorithm with a learning rate of 0.5, discount factor $\gamma = 0.9$, and fixed exploration probability $\epsilon = 0.5$. These settings were selected to balance convergence speed and computational tractability on the available hardware.

Table 2. Hyperparameters and neural network architecture used in the experiments.

Parameter	Value
Maximum number of resources	25
Maximum number of episodes	500
Learning rate (α)	0.5
Discount factor (γ)	0.9
Exploration probability (ϵ)	0.5 (fixed)
Optimizer	Adam
Loss function	Mean Squared Error (MSE)
Hidden layers	2
Neurons per hidden layer	64
Activation functions	ReLU (hidden), Linear (output)
Environment tasks	5

5.2. Discussion: Robustness Under Noisy Feedback

An important practical aspect concerns the robustness of the learning process when user feedback is noisy or delayed. For example, the satisfaction signal may be perturbed by stochastic noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ or provided with a delay of δ time steps. Such effects can slow convergence and increase reward variance.

Although a full experimental evaluation of noisy or delayed feedback is left for future work, preliminary reasoning suggests that Deep RL methods are expected to cope better with such imperfections compared to tabular Q-learning due to their generalization capacity. Investigating robustness under realistic QoE measurements will be an important direction for extending this proof-of-concept study.

Computational note. Although experiments with 10–25 resources may appear small compared to production-scale cloud systems, they were already computationally demanding on the hardware available to the authors. For example, training with 19 resources required more hours while training with 25 resources required approximately much more time. These practical constraints motivated our choice to limit the experimental scale in this study. Scaling to hundreds or thousands of resources would require a more powerful computing setup, which we plan to employ in future work.

Remark on experimental realism: It should be noted that our current evaluation relied on a synthetic Gym-based environment with artificially generated workloads and intents. While this setup was suitable for validating the proof of concept, it does not fully capture the complexities of production-scale systems. As part of future work, we plan to incorporate publicly available workload traces (e.g., the Google cluster trace) to assess the applicability and robustness of the proposed framework under realistic conditions.

6. Conclusions

We proposed two Q-learning-based approaches to map users' intentions into resource allocation decisions in an edge–cloud infrastructure: one leveraging a Q-Table and the other employing a neural network. The fundamental design elements of both the Reinforcement Learning (RL) and Deep-RL methodologies were described, including the state space, S, action space, A, and reward, r. We carried out simulation experiments focusing on storage requests with specific requirements, in the form of intents, in terms of the data size (capacity) and also regarding the cost (e.g., how much the user is willing to pay) and the security needs. The results demonstrate the capability of the proposed methods to perform infrastructure-agnostic resource allocation effectively in a way that is in accordance to users' actual intentions. Additionally, we investigated the impact of varying the number of resource types (resource levels) and the number of parameters considered (such as cost and

security) and evaluated the trade-off between executing optimal actions according to the Q-Table versus exploring alternative actions. Finally, a comparative analysis of the RL and Deep-RL approaches highlights the superior efficiency of the Deep-RL-based methodology.

Limitations and Future Work

The current evaluation focused on a single-user setting and infrastructures with up to 25 resources. This choice allowed us to validate our methodology in a tractable Gym-based environment and to isolate the core behaviour of intent-to-resource mapping. Although 10–25 resources may appear small, these experiments were already computationally demanding on the hardware available to the authors. For instance, with 19 resources, training time exceeded several hours, as mentioned, while with 25 resources, it increased to quite a few hours. Therefore, scaling to hundreds or thousands of resources requires a more powerful computing setup, which we plan to employ in future work. It is worth stressing that our experimental goal was precisely to demonstrate that even at relatively modest scales, the neural network-based approach begins to outperform classical tabular Q-learning. This provides strong evidence that the performance gap will widen further at larger scales, a hypothesis we intend to validate with stronger hardware and multi-user scenarios in subsequent studies.

Author Contributions: Methodology, D.K., P.S., A.V. and P.K.; Software, D.K., P.S., A.V. and P.K.; Writing—original draft, D.K.; Writing—review & editing, D.K., P.S., A.V. and P.K.; Supervision, P.K.; Project administration, P.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The simulation code used to generate the results of this study was developed by the authors. No external datasets were used. The code and experimental setup are available from the corresponding author upon reasonable request.

Acknowledgments: The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kretsis, A.; Kokkinos, P.; Soumplis, P.; Olmos, J.J.V.; Fehér, M.; Sipos, M.; Lucani, D.E.; Khabi, D.; Masouros, D.; Siozios, K.; et al. Serrano: Transparent application deployment in a secure, accelerated and cognitive cloud continuum. In Proceedings of the 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), Athens, Greece, 7–10 September 2021; pp. 55–60.
2. Kokkinos, P.; Margaris, D.; Spiliotopoulos, D. A Quality of Experience Illustrator User Interface for Cloud Provider Recommendations. In *HCI International 2022 Posters, Proceedings of the 24th International Conference on Human-Computer Interaction, HCII 2022, Virtual, 26 June–1 July 2022*; Springer: Cham, Switzerland, 2022; pp. 308–315.
3. Clemm, A.; Ciavaglia, L.; Granville, L.Z.; Tantsura, J. RFC 9315: Intent-Based Networking—Concepts and Definitions. RFC Editor. 2022. Available online: https://www.rfc-editor.org/rfc/rfc9315.html (accessed on 30 September 2025).
4. Mardian, R.D.; Suryanegara, M.; Ramli, K. Measuring Quality of Service (QoS) and Quality of Experience (QoE) on 5G Technology: A Review. In Proceedings of the 2019 IEEE International Conference on Innovative Research and Development (ICIRD), Piscataway, NJ, USA, 28-30 June 2019; pp. 1–6. [CrossRef]
5. Pang, L.; Yang, C.; Chen, D.; Song, Y.; Guizani, M. A survey on intent-driven networks. *IEEE Access* **2020**, *8*, 22862–22873. [CrossRef]
6. Leivadeas, A.; Falkner, M. VNF placement problem: A multi-tenant intent-based networking approach. In Proceedings of the 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 1–4 March 2021; pp. 143–150.
7. Hong, C.H.; Varghese, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *Acm Comput. Surv.* **2019**, *52*, 1–37. [CrossRef]

8. Sutton, R.S. Reinforcement learning: An introduction. In *A Bradford Book*; MIT Press: Cambridge, UK, 2018.
9. Kokkinos, P.; Varvarigos, A.; Konidaris, D.; Tserpes, K. Intent-Based Allocation of Cloud Computing Resources Using Q-Learning. In *International Symposium on Algorithmic Aspects of Cloud Computing*; Springer: Cham, Switzerland, 2023; pp. 184–196.
10. Al-Tamimi, A.; Lewis, F.L.; Abu-Khalaf, M. Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica* **2007**, *43*, 473–481. [CrossRef]
11. Carta, S.; Ferreira, A.; Podda, A.S.; Recupero, D.R.; Sanna, A. Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. *Expert Syst. Appl.* **2021**, *164*, 113820. [CrossRef]
12. Gao, Z.; Sun, T.; Xiao, H. Decision-making method for vehicle longitudinal automatic driving based on reinforcement Q-learning. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881419853185. [CrossRef]
13. Aihara, N.; Adachi, K.; Takyu, O.; Ohta, M.; Fujii, T. Q-learning aided resource allocation and environment recognition in LoRaWAN with CSMA/CA. *IEEE Access* **2019**, *7*, 152126–152137. [CrossRef]
14. Rezwan, S.; Choi, W. Priority-based joint resource allocation with deep q-learning for heterogeneous NOMA systems. *IEEE Access* **2021**, *9*, 41468–41481. [CrossRef]
15. Dab, B.; Aitsaadi, N.; Langar, R. Q-learning algorithm for joint computation offloading and resource allocation in edge cloud. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Washington, DC, USA, 8–12 April 2019; pp. 45–52.
16. Ning, Z.; Wang, X.; Rodrigues, J.J.; Xia, F. Joint computation offloading, power allocation, and channel assignment for 5G-enabled traffic management systems. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3058–3067. [CrossRef]
17. Kong, J.; Wu, Z.Y.; Ismail, M.; Serpedin, E.; Qaraqe, K.A. Q-learning based two-timescale power allocation for multi-homing hybrid RF/VLC networks. *IEEE Wirel. Commun. Lett.* **2019**, *9*, 443–447. [CrossRef]
18. Qiu, C.; Yao, H.; Yu, F.R.; Xu, F.; Zhao, C. Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5871–5883. [CrossRef]
19. Valkanis, A.; Beletsioti, G.A.; Nicopolitidis, P.; Papadimitriou, G.; Varvarigos, E. Reinforcement learning in traffic prediction of core optical networks using learning automata. In Proceedings of the 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), Sharjah, United Arab Emirates, 3–5 November 2020; pp. 1–6.
20. Alizadeh Javaheri, S.D.; Ghaemi, R.; Monshizadeh Naeen, H. An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing. *Computing* **2024**, *106*, 371–403. [CrossRef]
21. AlQerm, I.; Pan, J. Enhanced online Q-learning scheme for resource allocation with maximum utility and fairness in edge-IoT networks. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 3074–3086. [CrossRef]
22. Eshratifar, A.E.; Pedram, M. Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment. In Proceedings of the 2018 on Great Lakes Symposium on VLSI, Chicago, IL, USA, 23–25 May 2018; pp. 111–116.
23. Zheng, T.; Wan, J.; Zhang, J.; Jiang, C. Deep reinforcement learning-based workload scheduling for edge computing. *J. Cloud Comput.* **2022**, *11*, 3. [CrossRef]
24. Zeng, D.; Gu, L.; Pan, S.; Cai, J.; Guo, S. Resource management at the network edge: A deep reinforcement learning approach. *IEEE Netw.* **2019**, *33*, 26–33. [CrossRef]
25. Ramezani Shahidani, F.; Ghasemi, A.; Toroghi Haghighat, A.; Keshavarzi, A. Task scheduling in edge-fog-cloud architecture: A multi-objective load balancing approach using reinforcement learning algorithm. *Computing* **2023**, *105*, 1337–1359. [CrossRef]
26. Whang, S.E.; Roh, Y.; Song, H.; Lee, J.-G. Data collection and quality challenges in deep learning: A data-centric AI perspective. *VLDB J.* **2023**, 32, 791–813. [CrossRef]
27. Eimer, T.; Lindauer, M.; Raileanu, R. Hyperparameters in reinforcement learning and how to tune them. In I*nternational Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2023; pp. 9104–9149.
28. Abbas, K.; Afaq, M.; Ahmed Khan, T.; Rafiq, A.; Song, W.C. Slicing the core network and radio access network domains through intent-based networking for 5G networks. *Electronics* **2020**, *9*, 1710. [CrossRef]
29. Martini, B.; Gharbaoui, M.; Castoldi, P. Intent-based network slicing for SDN vertical services with assurance: Context, design and preliminary experiments. *Future Gener. Comput. Syst.* **2023**, *142*, 101–116. [CrossRef]
30. Velasco, L.; Signorelli, M.; De Dios, O.G.; Papagianni, C.; Bifulco, R.; Olmos, J.J.V.; Pryor, S.; Carrozzo, G.; Schulz-Zander, J.; Bennis, M.; et al. End-to-end intent-based networking. *IEEE Commun. Mag.* **2021**, *59*, 106–112. [CrossRef]
31. Mekrache, A.; Ksentini, A.; Verikoukis, C. Intent-Based Management of Next-Generation Networks: An LLM-centric Approach. *IEEE Netw.* **2024**, *38*, 29–36. [CrossRef]
32. Akbari, N.; Grundy, J.; Cheema, A.; Toosi, A.N. IntentContinuum: Using LLMs to Support Intent-Based Computing Across the Compute Continuum. *arXiv* **2025**, arXiv:2504.04429.
33. Chao, W.; Horiuchi, S. Intent-based cloud service management. In Proceedings of the 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 20–22 February 2018; pp. 1–5.

34. Kang, J.M.; Lee, J.; Nagendra, V.; Banerjee, S. LMS: Label management service for intent-driven cloud management. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 May 2017; pp. 177–185.
35. Liao, H.; Zhou, Z.; Kong, W.; Chen, Y.; Wang, X.; Wang, Z.; Al Otaibi, S. Learning-based intent-aware task offloading for air-ground integrated vehicular edge computing. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 5127–5139. [CrossRef]
36. Wu, C.; Horiuchi, S.; Murase, K.; Kikushima, H.; Tayama, K. Intent-driven cloud resource design framework to meet cloud performance requirements and its application to a cloud-sensor system. *J. Cloud Comput.* **2021**, *10*, 1–22. [CrossRef]
37. Tang, Q.; Xie, R.; Feng, L.; Yu, F.R.; Chen, T.; Zhang, R.; Huang, T. Siats: A service intent-aware task scheduling framework for computing power networks. *IEEE Netw.* **2023**, *38*, 233–240. [CrossRef]
38. Metsch, T.; Viktorsson, M.; Hoban, A.; Vitali, M.; Iyer, R.; Elmroth, E. Intent-driven orchestration: Enforcing service level objectives for cloud native deployments. *SN Comput. Sci.* **2023**, *4*, 268. [CrossRef]
39. He, L.; Qian, Z. Intent-based resource matching strategy in cloud. *Inf. Sci.* **2020**, *538*, 1–18. [CrossRef]
40. Ismail, A.A.; Khalifa, N.E.; El-Khoribi, R.A. A survey on resource scheduling approaches in multi-access edge computing environment: A deep reinforcement learning study. *Clust. Comput.* **2025**, *28*, 184. [CrossRef]
41. Zhou, G.; Tian, W.; Buyya, R.; Xue, R.; Song, L. Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *Artif. Intell. Rev.* **2024**, *57*, 124. [CrossRef]
42. Amazon, E. Amazon EC2 Instance Types. 2019. Available online: https://aws.amazon.com/ec2/instance-types (accessed on 30 September 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.