TYPE Review
PUBLISHED 31 May 2023
DOI 10.3389/fcomp.2023.1156064

Check for updates

OPEN ACCESS

EDITED BY
Chintha Tellambura,
University of Alberta, Canada

REVIEWED BY
Fan Jiang,
Xi'an University of Posts and Telecommunications, China
Ali Reza Heidarpour,
University of Alberta, Canada

*CORRESPONDENCE
Di Wu
✉ di.wu1@samsung.com

RECEIVED 01 February 2023
ACCEPTED 03 April 2023
PUBLISHED 31 May 2023

CITATION
Wu D, Li J, Ferini A, Xu YT, Jenkin M, Jang S, Liu X and Dudek G (2023) Reinforcement learning for communication load balancing: approaches and challenges.
Front. Comput. Sci. 5:1156064.
doi: 10.3389/fcomp.2023.1156064

COPYRIGHT
© 2023 Wu, Li, Ferini, Xu, Jenkin, Jang, Liu and Dudek. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Reinforcement learning for communication load balancing: approaches and challenges

Di Wu[1]*, Jimmy Li[1], Amal Ferini[1], Yi Tian Xu[1], Michael Jenkin[1], Seowoo Jang[2], Xue Liu[1] and Gregory Dudek[1]

[1]Samsung Artificial Intelligence (AI) Center, Montreal, QC, Canada, [2]Samsung Electronics, Seoul, Republic of Korea

The amount of cellular communication network traffic has increased dramatically in recent years, and this increase has led to a demand for enhanced network performance. Communication load balancing aims to balance the load across available network resources and thus improve the quality of service for network users. Most existing load balancing algorithms are manually designed and tuned rule-based methods where near-optimality is almost impossible to achieve. Furthermore, rule-based methods are difficult to adapt to quickly changing traffic patterns in real-world environments. Reinforcement learning (RL) algorithms, especially deep reinforcement learning algorithms, have achieved impressive successes in many application domains and offer the potential of good adaptabiity to dynamic changes in network load patterns. This survey presents a systematic overview of RL-based communication load-balancing methods and discusses related challenges and opportunities. We first provide an introduction to the load balancing problem and to RL from fundamental concepts to advanced models. Then, we review RL approaches that address emerging communication load balancing issues important to next generation networks, including 5G and beyond. Finally, we highlight important challenges, open issues, and future research directions for applying RL for communication load balancing.

KEYWORDS

reinforcement learning, wireless communication load balancing, 5G networks and beyond, WiFi network load balancing, real-world challenges

1. Introduction

Wireless communication networks have revolutionized the world, providing reliable high-bandwidth low latency communication through a variety of different technologies from WiFi networks to 5G and beyond. With the ongoing development and acceptance of these technologies, the number of mobile users and high data demanding mobile applications have also been increasing very quickly. As reported in Ericsson (2022), total global mobile data traffic was expected to reach 90 EB/month by the end of 2022 and to grow to around 115 EB/month by 2028. The number of mobile data users also shows a significant increase. In 2022 there were around 7.3 billion wireless data devices, a number that is expected to grow to over 9.2 billion devices by 2028.

Wireless cellphone network coverage is provided through a collection of radios and associated antennae that provide coverage over a given region in a given range of frequency bands. A group of antennae is ganged together to provide coverage over a sector over a small number of frequency bands, and sectors are grouped together to provide coverage over a wide range of orientations around a given point in space. A LTE base station in Canada, for example, might be comprised of three partially-overlapping sectors, equally spaced in orientation, with antennas at the 850 and 1,900 MHz frequencies. Generally, lower frequencies have a poorer data capacity but a longer range. A given UE is normally capable of communicating with more than one (frequency, sector pair) associated with a given base station (BS) and with more than one base station. Somewhat confusingly from a notational standpoint, the BS and the corresponding (frequency, sector pair) to which the user equipment (UE) is associated is known as the BS associated with the UE.

Each UE in the network communicates with the network through a single BS and is either active (it is actively consuming network resources), or it is inactive (it is said to be camping). As each individual UE consumes communication resources, and the capacity of the corresponding base station is limited, balancing the actual assignment of UEs to base stations is a fundamental problem in wireless network systems. The computational task associated with load balancing is complicated by a range of factors related to the dynamic nature of the network traffic including; the difficulty in estimating the future load associated with a given UE, the difficulty in estimating the future mobility of a given UE which impacts the availability of base stations to service the load from this UE, the predictability of new UEs requiring service, and the predictability of existing UEs departing the network.

Although it might be attractive to consider load balancing in a network as a global process, with access to all network properties including the number of UEs assigned to each base station along with their anticipated future load, this is not possible in practice. Practical load-balancing algorithms operate based on local properties of individual UEs (e.g., the UEs current signal strength associated with possible BSs) and BSs, rather than on global network properties, and balancing takes place at computational infrastructures which are associated with the BS. Practical algorithms must be computationally efficient as there may be 100's of millions of active UEs in a given network, and network data can be updated 100's to 1000's of times per second.

Early approaches to wireless network load balancing focused primarily on using rule-based methods, adaptive rule-based methods, and optimization based methods. To take but one example here, the A3-RSRP handover algorithm (see Hendrawan et al., 2019 and Figure 1), is based upon comparing Reference Signal Received Power (RSRP) between the currently serving BS and a competitor BS known as the target. Handover is controlled by three parameters; a hysteresis value, a cell individual offset ($\text{CIO}_{i,j}$) and a time to trigger timer. Different rule-based approaches can be used to set these parameters. More generally, the local load balancing process may use Reference Signal Received Quality (RSRQ) a measure of the quality of the underlying signal. RSSI and RSRQ are terminology related to LTE networks. Similar measures exist for 5G networks. For simplicity, we concentrate on load balancing for LTE (4G) networks here. We review load balancing in cellular networks in more detail in Section 2.

In recent years, machine learning techniques including reinforcement learning (RL) have been applied to the communication load balancing problem and achieved promising results. RL aims to learn a control policy via interaction with the environment (Sutton and Barto, 1998). But RL-based approaches also introduce their own complications. RL typically requires repeated interactions with the operating environment to learn an appropriate policy, and requires a well-crafted reward function in order to obtain the desired performance. How can we apply state of the art RL-based algorithms to wireless network load balancing so as to obtain the desired improvements in load balancing? The remaining of this paper is organized as follows. Section 2 describes the load balancing problem in cellular networks. Section 3 provides a review of RL, including the underlying concepts for both single agent RL and multi-agent RL. These concepts are expanded upon in Section 4 for single agent RL algorithms. Section 5 examines RL-based load balancing in cellular networks. Section 6 reviews RL-based load balancing for related problems in other wireless network systems. Section 7 reviews future challenges and suggests opportunities for RL in load balancing tasks.

2. Load balancing in cellular networks

The goal of load balancing in communication networks is to reduce the load imbalance among the cells[1], while minimizing any additional overhead on the service quality. In Section 2.1, we review standard load balancing mechanisms, in which various controllable parameters can be tuned to optimize load balancing. In Section 2.2, we review the metrics that are commonly used to assess the performance of a load balancing algorithm. It is worth noting that different papers may have different descriptions for load balancing mechanisms. Here, we only consider downlink communication scenarios as downlink communications is typically more significant than uplink.

2.1. Load balancing mechanisms

There are two main approaches to load balancing: Mobility Load Balancing (MLB), which redistributes the UEs through mobility mechanisms, such as handover and cell-reselection; and coverage-based load balancing, which extends or shrink the cell coverage by, for example, controlling the transmission power of the cells. Load balancing can be optimized by tuning the controllable parameters in these mechanisms. In the remainder of this section, we first briefly introduce the concept of MLB, and present the different control parameters for MLB and coverage-based load balancing.

MLB was introduced in LTE, and it is an important feature of self organizing networks (SON) (Jorguseski et al., 2014). MLB aims to offload UEs from an overloaded cell to a neighboring cell by adjusting handover and cell-reselection parameters. This mechanism is enabled by measurement reports from the UEs of their signal quality received from the serving and neighboring cells. These measurement reports are commonly described in terms of the Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ) reports, and are reported periodically within the network. Each wireless network specification includes

1 We use the term "cell" to refer to the smallest unit in a network that has an access point, to cover different base station structures studied in the literature, ranging from single to multiple sectors and/or sub-carriers.

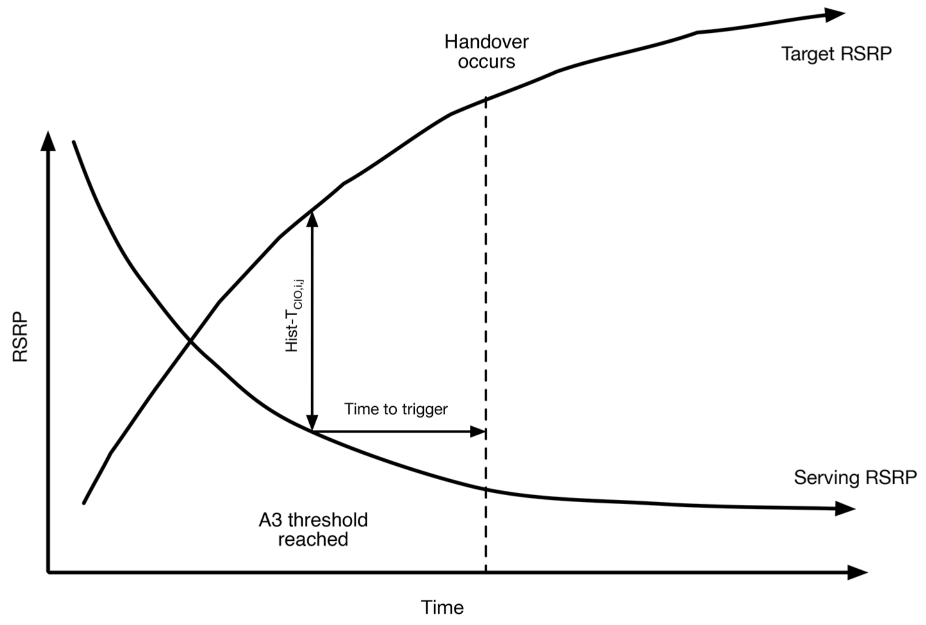

FIGURE 1
A3 RSRP rule-based handover algorithm. A UE receives RSRP reports from both the currently serving base station as well as a potential target base station. When the RSRP report for the target exceeds the currently serving station for a sufficiently long period of time (time to trigger) by a sufficient amount (hysteresis—HIST) less the cell individual offset (CIO$_{i,j}$) handover takes place. Setting the time to trigger, hysteresis and cell individual offset values controls the handover process.

a predefined set of measurement report mechanisms. In the LTE standard, for example, the UE reports events A1 through A6 which are defined for intra-LTE mobility and events B1 and B2 for inter-radio access technology (RAT) mobility (Addali et al., 2019).

The control parameters for MLB depends on the mode of the UE. Active UEs can be reassigned through a handover mechanism, while idle UEs are reassigned through cell-reselection. An UE is in idle mode when it is switched on, but does not have a Radio Resource Control (RRC) connection (Puttonen et al., 2009). Idle mode UEs receive limited or discontinuous reception, as a mean to save power consumption. For example, an idle UE wakes up occasionally to monitor incoming calls and measure for mobility, but it is soon allowed to sleep again, and it remains inactive for most of the time. Compared to UEs in active mode, idle mode UEs consume a negligible amount of network resources. Yet, they are expected to become active eventually to initiate requests and to increase the load of the cell that they are “camped on”. Therefore, proper distribution of the active and idle UEs help to balance the current and expected load, respectively.

2.1.1. MLB via handover for active UEs

A commonly used event in the load balancing literature is the A3 event illustrated in Figure 1. Using this event, handover is triggered when an UE’s signal strength (measured by its RSRP value) received from a neighboring cell is stronger than that from the serving cell (3GPP, 2011). An A3 event is triggered if the following condition is met

$$\text{A3: } X_j - X_i > Hyst - T_{\text{CIO}_{i,j}}, \tag{1}$$

where X_i and X_j are the signal quality measured from the serving and neighboring cell, respectively, $T_{\text{CIO}_{i,j}}$ is the Cell Individual Offset (CIO) between cells i and j, and $Hyst$ a hysteresis parameter that is used to discourage frequent handovers, and is usually a fixed value. By increasing $T_{\text{CIO}_{i,j}}$, the service area of cell i shrinks while that of cell j expands as UEs are more encouraged to be handed over from cell i to cell j.

Although A3 event-based handover is straightforward, it does not necessarily balance the load over the network. UEs are assigned to the cell with the highest signal strength, regardless of the capability of the given cell to serve the UE. Consequently, there exist a large number of other event-based handover mechanisms. For example, some load balancing approaches utilize the A4 event threshold (Munoz et al., 2011), which ensures a minimum level of signal quality (measured by RSRQ) from a neighboring cell, as in the following condition.

$$\text{A4: } X_j > Hyst + T_{\text{A4}_j}, \tag{2}$$

where T_{A4_j} is the controllable threshold for cell j. For example, Addali et al. (2019) uses the A4 event to select a neighboring cell for a potential early handover of edge UEs. It is also possible to use combinations of different events to drive the load balancing process. For example, Feriani et al. (2022) considers a combination of A2 and A5 events. Specifically, if the following two conditions are satisfied, then a handover is triggered.

$$\text{A2: } X_i + Hyst < T_{\text{A2}_i}, \tag{3}$$

$$\text{A5: } X_i + Hyst < T_{\text{A5}_{i,j}} \text{ and } X_j > Hyst + T'_{\text{A5}_{i,j}}, \tag{4}$$

where T_{A2_i}, $T_{\mathrm{A5}_{i,j}}$, and $T'_{\mathrm{A5}_{i,j}}$ are controllable thresholds. The A5 thresholds, $T_{\mathrm{A5}_{i,j}}$ and $T'_{\mathrm{A5}_{i,j}}$ are pairwise for each source-neighbor cell pair.

2.1.2. MLB via cell-reselection for idle UEs

Load balancing of idle mode UEs is realized by modifying the cell-reselection parameters, which control the selection of a "camping" cell. Cell-reselection is triggered if the quality of the RSRP/RSRQ measurements to the camping cell fall below a threshold. To select an alternative cell to camp on, CRS threshold values $T_{\mathrm{CRS}_{i,j}}$ are used. Each $T_{\mathrm{CRS}_{i,j}}$ is a pairwise threshold between i the cell currently camped on, and each potential alternative cell j. Once cell-reselection is triggered, the UE will rank the cells j that satisfy the condition (5)

$$X_j > T_{\mathrm{CRS}_{i,j}} \tag{5}$$

according to the signal quality and a preference parameter w_j associated with cell j. The cell with the highest ranking is selected as the "camping" cell. Feriani et al. (2022) control cell-reselection by tuning $T_{\mathrm{CRS}_{i,j}}$, while Wu et al. (2021) achieve this by tuning w_j.

2.1.3. Coverage-based load balancing

The data sent by cells is transmitted using radio signals with a controllable power level $P_i \in [P_{\mathrm{min}}, P_{\mathrm{max}}]$ dBm. Stronger transmission power results in higher signal quality measurements from the UEs at the cost of higher energy cost. While transmission power can be used for load balancing alone, it can also be used in concert with MLB parameters. As discussed in Alsuhli et al. (2021a), increasing the transmission power of a cell increases the signal quality of the non-edge UEs at that cell, but also increases the signal interference of the edge UEs at neighboring cells. While tuning P_i can be effective at controlling all the edge UEs of cell i simultaneously, MLB parameters can be used to control the UEs only at the common boundary between cells i and j.

2.2. Load balancing metrics

There exist a large number of metrics that seek to quantify the Quality of Service (QoS) provided to UEs in the network. Interestingly, providers are often more interested in their Quality of Experience (QoE), which is much more difficult to quantify. Let U be the set of UEs in the network. For simplicity, assume U to be a fixed set throughout the entire lifespan of the network. Let C be the set of cells in the network, and $U_c(\Delta t) \subseteq U$ be the set of UEs that are connected to cell $c \in C$ any time within a period of time in Δt, which can be the number of time transmission intervals (TTI).

2.2.1. Utilization based

The load in a cell is commonly expressed as a percentage of allocated physical resource blocks (PRB—the smallest resource block allocated by the network), or the PRB utilization ratio. The amount of PRB needed to allocate to a request depends on the signal-to-interference-plus-noise (SINR) ratio (Afroz et al., 2015). Let $\rho_c(\Delta t)$ be the average percentage PRB used during Δt. A straight-forward way to quantify the performance of load balancing is to measure the fairness of the utilization between cells. Examples of such metrics include the maximum load over all cells, average deviation and standard deviation of load over all cells. Minimizing these measures can serve as the objective function to optimize a load balancing algorithm. But there are other approaches. For example, in Asghari et al. (2021), the minimization of over-utilization is used instead by employing a penalty function that take effect when individual cell load exceeds a configurable threshold.

2.2.2. Throughput based

Throughput is the data transmitted per unit of time, expressed as bits per second (bps). More precisely, it is the rate at which packets arrive at their destination successfully, without counting those lost in transit. This metric quantifies the speed of the network. Let $f_{u,c}(\Delta t)$ be the total number of packets transmitted to the UE u from the cell c, and $d_{u,c}(\Delta t)$ be the total delay experienced by the UE u when receiving packets from cell c during Δt. Then the throughput is defined as

$$\mathrm{Tput}_{u,c}(\Delta t) = \frac{f_{u,c}(\Delta t)}{d_{u,c}(\Delta t)}, \tag{6}$$

The throughput is 0 when $f_{u,c}(\Delta t) = 0$. As cell congestion causes more delays, the throughput from a cell decreases when the load on that cell approaches its maximum capacity. The sum or average throughput over all UEs and cells is considered as the overall performance of the network, and can be used as the objective to maximize for load balancing.

Similar to utilization-based metrics, fairness between cells can be measured in terms of the minimum and the standard deviation of throughput from the cells. The throughput from a cell c can be expressed as:

$$\mathrm{Tput}_c(\Delta t) = \frac{\sum_{u \in U_c(\Delta t)} f_{u,c}(\Delta t)}{\sum_{u \in U_c(\Delta t)} d_{u,c}(\Delta t)}, \tag{7}$$

and $\mathrm{Tput}_c(\Delta t) = 0$ if $|U_c(\Delta t)| = 0$. A penalty on cells with low throughput can also be formulated as in Wu et al. (2021), that has a similar functionality as a penalty on over-utilized cells.

2.2.3. Other performance metrics

The signal to interference plus noise ratio (SINR) and the channel quality indicator (CQI) can be used to construct metrics that are more tailored to the QoE. Examples include the average SINR and the ratio of UEs out of coverage. The CQI is a discretized measurement of the signal quality as measured by the UE. Low values indicate that the UE is out of coverage, and higher value correspond to better signal quality, and hence better modulation and coding scheme (MCS) assignment (Alsuhli et al., 2021a; Aboelwafa et al., 2022). In Mwanje and Mitschele-Thiel (2013), the number of unsatisfied UE is used to evaluate the performance, where an UE is unsatisfied if it is served with fewer PRB than requested, resulting in a data rate lower than the Guaranteed

Bit Rate (GBR). The requested PRB to match a certain data rate depends on the SINR.

Other key performance indicators that are often used in the load balancing literature and that may not be accountable in utilization-based or throughput-based metrics include the call drop rate, call block rate, handover failure ratio, ping-pong ratio and outage ratio. An admission control mechanism in the network can block calls or handovers based on the availability of the resource (Kwan et al., 2010). Intuitively, overloaded cells have a higher call block rate, and UEs handed over to those cells are more likely to be blocked and have their calls dropped, or handed back to the original cell, potentially resulting in a ping-pong effect. When a cell cannot serve its connection due to the lack of resource or low SINR, this can be measured by the outage ratio. Some settings of the load balancing parameters can also cause inappropriate configuration of cell coverage, resulting in the deterioration of the aforementioned performance indicators. Given the range of different indicators, the optimization of load balancing is often approached from a multi-objective perspective.

3. RL preliminaries

This section provides a brief overview of single agent and multi-agent RL. We present the mathematical frameworks commonly used to model sequential decision making for single and multiple agents. For a more detailed review of RL the interested reader is directed to one of the classical texts on RL (e.g., Sutton and Barto, 1998) or a recent review paper such as Nian et al. (2020).

Reinforcement learning is commonly formalized as a Markov Decision Process (MDP). Assuming the environment is fully observable, a MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \kappa_0)$, where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; $\mathcal{P}(s'|s,a)$ is a transition probability function which outputs the probability of transiting to s' from the current state s after executing the action a; $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is a scalar reward function; $\gamma \in [0,1]$ is the reward discount factor; and κ_0 is the initial state distribution (Puterman, 1994).

In RL, the agent aims to learn a policy π, a mapping from states to actions, such that the expected cumulative discounted reward $\mathcal{J}(\pi)$ of the policy in the MDP is maximized:

$$\begin{aligned}\pi^* &= \underset{\pi \in \Pi}{\arg\max}\, \mathcal{J}(\pi) \\ &= \underset{\pi \in \Pi}{\arg\max}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) \,\middle|\, a_t \right. \\ &\left. \sim \pi(a_t|s_t), s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t), s_0 \sim \kappa_0 \right]; \qquad (8)\end{aligned}$$

where π^* is the optimal policy and Π is the set of all possible policies. The learning objective in (8) assumes an *infinite-horizon* MDP where the agent continuously interacts with the environment. This is in contrast to *episodic* MDP where the interactions between the agent and the environment terminate after a bounded number of steps. Note that the learning objective can be adjusted to the episodic case by considering a finite horizon.

Instead of maximizing $\mathcal{J}(\pi)$ that is a function of the policy, one can consider other performance objectives such as the value function $V^\pi: \mathcal{S} \mapsto \mathbb{R}$ that measures the expected accumulated reward starting from a particular state and following the policy π:

$$\begin{aligned}V^\pi(s) = \mathbb{E}&\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) \,\middle|\, a_t \sim \pi(a_t|s_t), s_{t+1} \right. \\ &\left. \sim \mathcal{P}(s_{t+1}|s_t, a_t), s_0 = s \right] \qquad (9)\end{aligned}$$

The state-action value function (i.e., the Q function) is defined as the expected accumulated reward starting from a particular state, executing and action a and following the policy π thereafter as in:

$$\begin{aligned}Q^\pi(s,a) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) \,\middle|\, a_t \sim \pi(a_t|s_t), s_{t+1} \right. \\ &\left. \sim \mathcal{P}(s_{t+1}|s_t, a_t), s_0 = s, a_0 = a \right] \qquad (10) \\ &= \mathbb{E}[\mathcal{R}(s,a,s') + \gamma V^\pi(s') | s' \sim \mathcal{P}(s'|s,a)] \qquad (11)\end{aligned}$$

MDPs assume that the agent has full access to the environment states. However, this assumption is not always realistic. Partially Observable MDPs (POMDP) (Åström, 1965) are a generalization of MDPs where the agent has only access to a partial observation of the state. Formally, a POMDP consists of a tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \zeta, \kappa_0)$ where $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma$ and κ_0 are as defined above, $\mathcal{O}$ is the observation space and $\zeta: \mathcal{S} \times \mathcal{A} \times \mathcal{O} \mapsto [0,1]$ denotes the probability distribution over observations given a state s and an action a. To overcome the partial observability, the agent maintains an estimate of the environment state b, often called the *belief state*. The belief state is defined as the conditional probability distribution over all possible states given the history of past actions and observations. If the dynamics of the POMDP are known, the belief states can be updated using Bayes' rule. Consequently, the agent aims to learn a policy π that maps its current belief state to action while keep updating the belief states when new observations are received.

4. RL algorithms

The previous section reviewed the key formalisms used in RL. Here, we briefly review key RL algorithms that are relevant to our discussion. We focus on model-free RL, which is the predominant RL approach used in the load balancing literature. This type of RL algorithm aims to directly output control actions based on the state, without building an explicit system dynamics model. Model-free RL methods are typically categorized into two families: value- and policy-based algorithms. In recent years, reinforcement learning, especially deep reinforcement learning has also been applied to solve different types of real-world problems (Wu, 2018; Fu et al., 2022b; Li et al., 2022).

4.1. Value-based RL

Value-based algorithms estimate the agent's value function expressed in Equation (9) or the state-action value function in Equation (10). The agent's policy is then computed *greedily* with respect to the approximated value function such that the selected action is the one that maximizes the approximated value function at a given state s.

The Value Iteration algorithm (Sutton and Barto, 1998), a well-known dynamic programming method, approximates the optimal Q^* by iteratively applying the Bellman optimality operator $[BQ](s,a) = \mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{P}(s'|s,a) \max_{a'} Q^*(s',a')$. However, model-free RL assumes that the system's dynamics (i.e., the transition and/or reward functions) are unknown, hence it is not possible to evaluate the Bellman operator for this category of RL. To overcome this limitation, Temporal Difference (TD) based methods update the estimated state-action values $\hat{Q}$ as follows:

$$\hat{Q}(s,a) = (1-\alpha)\hat{Q}(s,a) + \alpha\big[\mathcal{R}(s,a) + \gamma \max_{a'} \hat{Q}(s',a')\big]; \quad (12)$$

where α is a learning rate. To evaluate (12), the agent collects experience samples through interactions with its environment. Depending on the policy used to collect these samples, TD methods can be *on-policy* or *off-policy*. The State-Action-Reward-State-Action (SARSA) algorithm (Sutton and Barto, 1998) is a well-known on-policy TD method where samples $\{(s,a,r,s',a')\}$ are collected by acting greedily with respect to the learned Q function. Alternatively, the famous Q- algorithm (Watkins and Dayan, 1992) is an off-policy TD algorithm where any behavior policy can be used to generate experiences. The Q-learning algorithm is the foundation of the modern value-based algorithms where $\hat{Q}$ is learned by minimizing the Bellman error defined as:

$$\mathcal{L}_Q = \sum_{(s,a,r,s')} ||\hat{Q}(s,a) - (\mathcal{R}(s,a) + \gamma \max_{a'} \hat{Q}(s',a'))||^2. \quad (13)$$

Modern Q-learning algorithms represent the Q function as a deep neural network with parameters θ. The pioneering Deep Q-Network (DQN) enabled deep Q learning (Mnih et al., 2013). In this work, two main techniques are introduced to stabilize the learning using deep neural networks as function approximators. To avoid using correlated samples, an experience replay buffer $D = \{(s,a,r,s')\}$ is used and at each iteration, the Q function is updated using experiences sampled from the replay buffer D. DQN also uses a target network $\bar{Q}$ to evaluate the targets in the Bellman error (13). The target network parameters are a copy of the Q function parameters θ and are updated periodically with the most recent values. Taking into consideration these modifications, the new expression of the Bellman error becomes:

$$\mathcal{L}_{DQN} = \sum_{(s,a,r,s')\sim D} ||\hat{Q}(s,a) - (\mathcal{R}(s,a) + \gamma \max_{a'} \bar{Q}(s',a'))||^2. \quad (14)$$

Several variants of the DQN algorithm have been proposed in the literature to overcome the instability and maximization bias of the original work. For instance, double DQN (DDQN) (Van Hasselt et al., 2016) proposed to use two target networks $\bar{Q}_1$ and $\bar{Q}_2$ with different parameters to decouple the action selection from the action evaluation in the targets (i.e, $\mathcal{R}(s,a) + \gamma \bar{Q}_2(s', \arg\max_{a'} \bar{Q}_1(s',a'))$). Furthermore, prioritized experience replay is introduced to improve the converge guarantees by sampling rare or task-related experiences more frequently than the redundant ones (Schaul et al., 2015). Another variant, dubbed dueling DQN (Wang et al., 2016), computes the Q-function as the difference between a value network and a state-dependent action advantage network.

4.2. Policy-based RL

Unlike value-based RL methods described above where the optimal policy is computed greedily with respect to the Q function, policy-based RL algorithms search for the optimal policy directly. The optimal policy is obtained by maximizing the agent's expected cumulative discounted rewards as in (8). The policy is often represented as a function approximator (e.g., a deep neural network) with learnable parameters ϕ. The seminal work by Sutton et al. (2000) introduced the family of *Policy Gradient* (PG) methods that learn the optimal policy parameters by applying gradient ascent on the objective $\mathcal{J}$. More specifically, the policy gradients are estimated using sampled trajectories or rollouts collected under the current policy as in:

$$\nabla_\phi J(\pi_\phi) = \mathbb{E}_{\tau\sim\pi_\phi}\left[\sum_{t=0}^{H} \nabla \log \pi_\phi(a_t|s_t) Q^{\pi_\phi}(s_t,a_t)\right]; \quad (15)$$

where $\tau = \{s_t, a_t, r_t\}_{t=0}^{H}$ is a trajectory, H is the length of the trajectories and Q^{π_ϕ} is the state-action value function under the policy π_ϕ.

In the well-known REINFORCE algorithm (Williams, 1992), the Q^{π_ϕ} function is defined as the *rewards-to-go* [i.e., $Q^{\pi_\phi}(s_t,a_t) = \sum_{k=t}^{H} \mathcal{R}(s_k,a_k)$] where the expected return at a state-action pair (s_t,a_t) is the sum discounted rewards from time t until the end of the trajectory. The PG methods are unbiased but they are characterized by high variance. Thus, a state-dependent baseline is subtracted from the Q^{π_ϕ} to reduce the variance and keep the gradient estimates unbiased. A commonly used baseline is the state value function V^{π_θ}.

Actor-critic methods are an extension of the PG methods where the Q^{π_ϕ} function is learned in addition to the policy. Hence, actor-critic methods often learn two models. The critic $Q_\theta^{\pi_\phi}$ parameterized by the parameters θ approximates the state-action value function. Subsequently, the actor or the policy π_ϕ is updated based on the learned critic. Consequently, the actor-critic methods bridge the PG methods and the value-based ones. Note that the critic is not restricted to the state-action value function. For instance, Advantage Actor-Critic (A2C) and Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016) approximate the advantage function A^{π_θ}.

To improve the convergence of REINFORCE and its variants (e.g., A3C, A2C), the Trust Region Policy Optimization (TRPO) algorithm was proposed to constrain the difference between the new updated policy and the old one below a certain threshold (Schulman et al., 2015). Although this additional constraint avoids destructive policy updates, TRPO requires additional computation since it relies on second-order optimization. To overcome this issue, Proximal Policy Optimization (PPO) presented a first-order alternative to TPRO that is easier to implement and has similar performance to the second-order method (Schulman et al., 2017). There are two variants of PPO: PPO-penalty and PPO-clip. We will focus on PPO-clip because it is the variant most commonly used. PPO-clip introduced a new surrogate loss function that performs clipping to penalize the agent when the new policy moves far from the old one:

$$L_{\mathrm{PPO}}(\phi) = \min_{\phi} \left(\frac{\pi_\phi(a|s)}{\pi_{\phi_k}(a|s)} A^{\pi_{\phi_k}}(s,a), g(\delta, A^{\pi_{\phi_k}}(s,a)) \right); \qquad (16)$$

where

$$g(\delta, A) = \begin{cases} (1+\delta)\, A & \text{if } A \geq 0 \\ (1-\delta)\, A & \text{if } A < 0. \end{cases} \qquad (17)$$

δ is a hyperparameter to determine how far the new policy can move far from the old policy.

The policy in PG methods is usually modeled as a probability distribution with learnable parameters (i.e., the mean and the standard deviation of a Gaussian distribution). Hence, the policy learned by the PG methods is stochastic. As an alternative, the Deterministic Policy Gradient (DPG) method by Silver et al. (2014) and its deep version Deep DPG (DDPG) by Lillicrap et al. (2015) adopt a deterministic policy that outputs the action with the highest state-action value. Several extensions of the DDPG methods have been developed to improve learning speed and performance. As an example, Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) introduced several modifications such as the clipped double Q-learning to reduce the overestimation bias. Another example is Soft Actor-Critic (SAC) (Haarnoja et al., 2018) that added a maximum entropy term to improve exploration. In general, value-based reinforcement learning algorithms are more suitable to deal problems with discrete control actions while policy-based methods are more suitable to deal with problems with continuous control actions (Fu et al., 2022a,b).

5. Reinforcement learning-based load balancing in cellular networks

The use of RL for load balancing in cellular networks has become increasingly popular in recent years, driven both by the growth in wireless traffic, and the rapid advancement of RL and deep learning. Figure 2 illustrates the overall process. The RL agent monitors the state of the communication network. When it observes load imbalance, it selects the values for the load balancing parameters, such as CIO, to redistribute the UEs such that the appropriate load balancing metric(s) are optimized.

The ability of RL-based load balancing system to output control actions without the need for wireless network engineers to develop a system model is highly attractive in the context of cellular networks. Manually modeling system dynamics in cellular networks requires extensive domain knowledge, substantial human effort, and is often highly challenging given the complexity of the real world environments and the dynamic and ever changing nature of traffic patterns.

In this section, we present the prominent lines of work on RL for load balancing. Table 1 lists these lines of work, and presents key properties regarding the MDP and RL technique for each related group of papers. The following subsections highlight overarching themes seen across the literature, first in terms of the MDP problem formulation and then in terms of RL methods.

5.1. MDP formulations

RL methods require the specification of a Markov Decision Process (MDP) as part of the problem formulation. This involves the specification of the state space, the action space, and the reward function. In the cellular load balancing literature, the state space and the reward function are typically based on key performance indicators (KPIs) that characterize the network performance. Examples of commonly used KPIs include throughput, active UEs, and PRB usage, and call block rate. There is often overlap between the KPIs used to compute the state and the KPIs used to compute the reward. This is because the KPIs useful for evaluating load balancing performance can also be useful for informing the next action that should be taken. For example, Wu et al. (2021) uses cell throughput to compute both state and reward: the throughput of each cell is included in the state vector, and the minimum throughput among cells is used as the reward.

The specific KPIs used to compute state and reward varies greatly between the different lines of work, since system performance and evenness of load distribution can be measured by many different KPIs. Some papers gauge load balancing performance based on the performance of the overall network. For example, Musleh et al. (2017) computes the average cell throughput as part of the reward without capturing the difference in load between cells. A contrasting example is the work of Feriani et al. (2022), which explicitly captures load distribution by using the standard deviation of cell load as part of the reward.

Unlike state and reward, the choice of action space is relatively consistent among the various lines of work. Cell individual offset (CIO) is used in almost all of the papers we review. An advantage of this approach is that existing base stations already support A3 handovers based on CIO values, and so it is natural to integrate into existing infrastructure. Some authors such as Wu et al. (2021) directly use CIO values in the action space, while other authors such as Mwanje et al. (2016) use CIO increment as the action space, in which case the output of the RL agent is an increment that is added to the current CIO value. Since controlling CIO only affects actively transmitting UEs, a number of recent papers (i.e., Wu et al., 2021; Feriani et al., 2022) use both CIO and cell reselection (CRS) thresholds in the action space, so that both active and idle UEs are considered during load balancing. Transmission power is used occasionally in the action space (Munoz et al., 2013; Musleh et al., 2017; Aboelwafa et al., 2022), which allows the RL agent to affect the coverage area of various cells. In comparison with controlling CIOs and CRS thresholds, directly controlling transmission power can be more risky since changes in transmission power may result in coverage holes.

Notably, an under-explored approach is to train the RL agent to directly assign UEs to cells rather than relying on the handover and cell reselection processes. The work of Ma et al. (2022) is the only work in our review that explores this. This is a challenging approach since the action space is much larger and dependent on the number of UEs. For simplicity, the number of UEs is assumed to be fixed. However, direct UE-cell assignment provides the RL system the greatest level of control, and is a promising direction for future work.

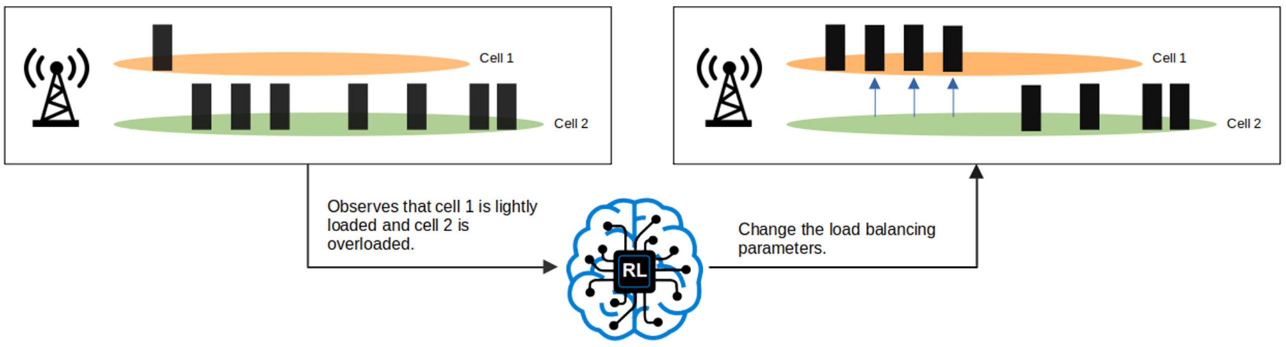
Cell 1
Cell 2
Cell 1
Cell 2
Observes that cell 1 is lightly loaded and cell 2 is overloaded.
RL
Change the load balancing parameters.

FIGURE 2
Illustration of load balancing between two cells using RL. When the RL agent observes network states that indicate load imbalance, it outputs the action that corresponds to the load balancing parameters in the network (e.g., CIO values) which allows redistribution of the UEs from the overloaded cell to the lightly loaded cell.

TABLE 1 **Recent papers applying RL to wireless network load balancing.**

	State	Action	Reward	RL method
Munoz et al. (2011), Munoz et al. (2013)	Call block rate, outage ratio, CIO, transmit power	CIO increment, transmit power increment	Call block rate, outage ratio	Q-learning
Mwanje and Mitschele-Thiel (2013), Mwanje et al. (2016)	PRB utilization, fraction of edge users	CIO increment	Source reduction and target cell increase in PRB utilization	Q-learning
Musleh et al. (2017)	Transmission power	Transmission power	Average SINR, average cell throughput, call drop rate, call block rate	Q-learning
Xu et al. (2019a), Xu et al. (2019b)	PRB utilization, fraction of edge users	CIO	Min of max PRB utilization	Off-policy actor critic
Attiah et al. (2020), Alsuhli et al. (2021a), Alsuhli et al. (2021b), Aboelwafa et al. (2022)	PRB utilization, downlink throughput, active UE count, MCS utilization	CIO, transmission power, MIMO on/off	Total throughput, number of blockage events, deviation of resource block utilization, total throughput minus expected throughput of users without coverage, number of cells with MIMO on	DQN (Mnih et al., 2013), DDQN (Van Hasselt et al., 2016), TD3 (Fujimoto et al., 2018)
Asghari et al. (2021)	Utilization level	CIO	Penalty on unsatisfied load	Q-learning, SARSA
Kang et al. (2021), Wu et al. (2021), Feriani et al. (2022), Li et al. (2022)	Throughput, PRB utilization, active UE count	CIO, CRS	Minimum, average, and deviation in throughput among cells; number of cells with throughput below a threshold	SAC (Haarnoja et al., 2018), PPO (Schulman et al., 2017)

5.2. RL-based methods for load balancing

As discussed in Section 4, model-free RL is the predominant RL approach across the load-balancing literature. Notably, all papers listed in Table 1 use model-free RL. Both value-based methods and policy-based methods have been explored. We first discuss load balancing techniques centered around value-based Q-learning, followed by those that rely on policy-gradient methods.

5.2.1. Q-learning for load balancing

The value-based Q-learning method is particularly popular throughout the literature, especially in earlier work. Q-learning typically requires discrete state and action spaces. However, network metrics and load balancing control knobs are often continuous, which necessitates a discretization scheme when applying Q-learning. The state is often discretized into regular intervals, and actions are often selected from a pre-defined set (i.e., Mwanje et al., 2016; Musleh et al., 2017).

Munoz et al. (2011) and Munoz et al. (2013) combine Q-learning with a fuzzy logic controller (FLC), which translates between continuous values into discrete semantic labels (i.e., high, medium low). Continuous states are fuzzified into discrete values, which are used as input to the RL agent; discrete actions produced by the RL agent are defuzzified into continuous values and then passed to the system. In this way the RL agent outputs incremental adjustments to the CIO values and transmit power to reduce service disruptions.

Mwanje and Mitschele-Thiel (2013) and Mwanje et al. (2016) also use Q-learning to control incremental adjustments to CIO values. However, unlike the typical RL setup where the learned agent makes decisions at every step, the agent is invoked conditionally when a cell is overloaded. The agent is rewarded for successfully reducing source cell load and increasing the load of neighboring target cells.

Musleh et al. (2017) also adopts the strategy of conditionally invoking a Q-learning agent when cell overload is detected. They focus on a heterogeneous networks consisting of high-powered macrocells and low-powered femtocells. When a macrocell is overloaded, neighboring femtocells independently select power levels using an RL agent, allowing them to increase coverage and take load off of the macrocell.

Asghari et al. (2021) is another work that embeds a Q-learning agent within a larger control flow framework. The system first identifies overloaded source cells and lightly loaded neighboring cells. Then an RL agent is used to select increments for the CIO values until the source cell is no longer overloaded.

Deep Q-Networks relaxes the need for state discretization by using deep network to approximate the qQfunction, leaving only the action space to be discretized. The work of Attiah et al. (2020) and Alsuhli et al. (2021b) are, respectively, based on DQN and the double DQN (DDQN) method proposed by Van Hasselt et al. (2016). Although both papers use the RL agent to control CIO values, Attiah et al. (2020) simplifies the problem by considering only a single CIO value for all neighboring cells, whereas Alsuhli et al. (2021b) considers separate CIO values for each neighboring cell. In addition to CIO, Alsuhli et al. (2021b) also includes transmission power in the action space. Alsuhli et al. (2021a) uses TD3, a policy gradient RL algorithm proposed by Fujimoto et al. (2018), to address a similar problem setup. The policy gradient approach allows for continuous action spaces, and enables the system to output continuous control for CIO and transmission power. Aboelwafa et al. (2022) combines both value-based and policy-based approaches in a single framework: DDQN is used to make discrete on/off decisions for MIMO features, and TD3 is used to control CIO and transmission power.

5.2.2. Policy-gradient methods for load balancing

Policy-based methods are gradually gaining popularity among more recent papers, since they are naturally compatible with continuous state and action spaces. This avoids the need to engineer a sensible discretization scheme, and can potentially avoid undesired human bias in this process. An additional trend in recent papers is the integration of RL with other learning techniques such as clustering (i.e., Xu et al., 2019b), hierarchical learning (i.e., Kang et al., 2021), meta-learning (i.e., Feriani et al., 2022), and knowledge distillation (i.e., Li et al., 2022). These additional techniques complement the core RL method and address key challenges such as scalability, adaptability, and model generalization.

Xu et al. (2019a) proposes a policy gradient method that uses deep networks to approximate an actor and a critic, much like the work of Lillicrap et al. (2015). The RL agent directly outputs continuous CIO values to minimize the maximum load among cells. Xu et al. (2019b) extends the approach by introducing a two-layer architecture that handles large-scale deployments. The top layer clusters base stations, using overloaded base stations as cluster centers; the bottom layer uses an RL agent to perform intra-cluster load balancing.

Wu et al. (2021) emphasizes the data efficiency challenge for deep reinforcement learning and proposes the first solution to improve the data efficiency in learning an RL-based load balancing solution for idle mode users. The soft actor-critic (SAC) method of Haarnoja et al. (2018) is used to optimize CRS values. The authors proposed to first identify a suitable source policy and then do shallow finetuning on the target domain which can help to significantly improve data efficiency.

In Kang et al. (2021), the authors propose a hierarchical policy learning framework for load balancing for both active users and idle users. Proximal policy optimization (PPO) of Schulman et al. (2017) is the RL algorithm used for this work. A higher level PPO policy determines CIO control actions for active users and a lower level PPO policy determines the CRS control actions for idle users.

Feriani et al. (2022) addresses conflicting objectives that arise during load balancing. As an example, the authors show that increasing the throughput of the most under-utilized cell can conflict with reducing the standard deviation of throughput among cells. In other words, the RL control policy should be trained to accommodate the preferences toward various key performance indicators (KPIs). The authors propose an approach based on meta-learning, inspired by the work of Finn et al. (2017), where a generic model is trained with the ability to quickly adapt to changes in KPI preferences. This work uses PPO as the core RL method, which controls both CIO and CRS values.

Li et al. (2022) focuses on the construction of a concise RL policy bank that can cope with a large variety of traffic patterns. This approach aims to improve the scalability of deploying RL policies across a many sites. In the next section, we describe this work in more detail as an example to showcase the use of RL for load balancing.

5.3. Sample application of RL to network load balancing

Here, we present the work of Li et al. (2022) as an example to showcase the use of RL for communication load balancing. In addition to presenting a RL formulation for load balancing, this work also investigates practical challenges regarding the scalable deployment of RL policies across a large geographic region. Notably, training a single RL policy for every base station across a large geographic region is often not feasible due to traffic variations, while maintaining separate RL policies for every base station overlooks similarities between some of the base stations. Thus, the authors propose a clustering-based approach: a set of task-specific RL policies are first trained on a diverse set of traffic scenarios or tasks; then, knowledge distillation is used to recursively merge together similar RL policies that exhibit similar behaviors. This approach produces a concise policy bank and reduces the overhead of maintaining a large number of policies when deploying across a large geographic region.

The MDP formulation used by Li et al. (2022) to train task-specific policies is shown below. PPO is used to train RL policies using this MDP.

- Each state vector $s_t \in \mathcal{S}$ consists of the number of active UEs in each cell, the PRB utilization of each cell, and the average throughput of each cell.

- Each action vector $a_t \in \mathcal{A}$ consists of the CIO values $T_{\text{CIO}_{i,j}}$ that trigger active UE handover as described in Section 2.1, as well as thresholds $\beta_{i,j}$ and $T_{\text{CRS}_{i,j}}$ such that cell reselection between cells i and j is triggered when $\text{RSRP}_i < \beta_{i,j}$ and $\text{RSRP}_j > T_{\text{CRS}_{i,j}}$.
- The reward $r_t \in \mathbb{R}$ is a weighted average of several metrics presented in Section 2.2, including the minimum throughput Tput_{min} among cells, the average Tput_{avg} throughput among cells, the standard deviation Tput_{std} among cells and the number of cells $\text{Tput}_{<\chi}$ whose throughput is below a threshold χ. Specifically, $r_t = \mu_1 \text{Tput}_{avg} + \mu_2 \text{Tput}_{min} + \mu_3 (1 + \text{Tput}_{sd})^{-1} + \mu_4 (C - \text{Tput}_{<\chi})$, where C is the number of cells. Note that under this formulation, maximizing the reward minimizes Tput_{sd} and $\text{Tput}_{<\chi}$. The coefficients μ_i are selected using grid search to maximize performance.

Once a set of task-specific RL policies are trained, the policies are clustered recursively. At each clustering step, the two most similar policies are identified and merged. The similarity between two policies is measured using the L2 distance between action vectors produced by the policies given the same input state. Policies are merged using knowledge distillation, where a student policy is trained to mimic the output of two teacher policies, as proposed by Hinton et al. (2015).

The clustering method is compared with five baselines listed below. When evaluating the clustering-based methods, the policy of each cluster is evaluated on all the tasks that belong to its cluster.

- Fixed: The same hand-crafted control parameters are used for all tasks at all time steps.
- J-Multitask: One RL agent is trained jointly on all tasks simultaneously. This is a form of multitask learning similar to the work of Pinto and Gupta (2017), which capitalizes on positive transfer between tasks.
- Task-specific: A separate RL agent is trained for each task and no clustering is performed. Since this approach does not require policies to generalize across multiple tasks, the performance of this method can be considered as an upper bound for the clustering methods.
- EM: An approach proposed by Ackermann et al. (2021) that clusters policies using an expectation maximization(EM) scheme: In the expectation step, each task is assigned to the cluster whose policy maximizes performance on the task. In the maximization step, the RL policy of each cluster is trained jointly on all tasks in its cluster using J-Multitask.
- Kmeans: A method in which the Fixed method is used to interact with the various tasks, and the states collected from these interactions are used to cluster the tasks via Kmeans. For each cluster, we train one RL policy for all tasks in the cluster using J-Multitask.

All methods are evaluated using a system-level simulator for 4G/5G communication networks. The simulator's parameters are tuned to mimic real-world traffic data at various sites to emulate different tasks. Two sets of traffic scenarios are used: the "Hex 1" scenario set consists of 10 relatively simple scenarios in which a single base station is simulated; the "Hex 7" scenario set consists of 32 relatively congested traffic scenarios in which a center base station is surrounded by six additional base stations that introduce interference. In all cases, each base station has three sectors, and the RL agent is expected to perform inter-cell load balancing within the one of the sectors in the center base station.

Experimental results are shown in Figure 3 and Table 2. Since the traffic scenarios vary greatly in traffic load, the attainable performance (i.e., absolute values of rewards and metrics) is highly dependent on the traffic scenario on which a method is evaluated. Thus, all results are expressed as relative values with respect to the Fixed method. Figure 3 shows the spread of absolute improvement in reward across tasks and Table 2 shows the percentage improvement in individual metrics that make up the reward. Overall, the proposed clustering method is able to outperform the other clustering methods, and closely match the performance of the task-specific method with only a small number of policies. This demonstrates that a concise policy bank constructed with the proposed method can generalize across diverse traffic scenarios.

6. RL-based load balancing in other communication domains

6.1. Load balancing for mobile edge computing

With the increase in the number of IoT devices, Mobile Edge Computing (MEC) has emerged as an infrastructure to provide cloud computing services at the network's edge, thereby enabling low latency and high bandwidth applications. IoT devices can benefit from resources associated with edge servers and send edge servers requests when the IoT device lack the processing capacity to execute a specific task locally. To increase capacity and decrease processing delays, load balancing plays a crucial role in MEC networks.

Load balancing between mobile devices and edge servers involves offloading tasks or computations from the devices to the edge servers. This is known in the literature as task or computation offloading. Depending on the task offloading strategies of the devices, some edge servers may receive more requests than others which can result in an unbalanced load at the edge server level. In this context, load balancing between edge servers involves evenly distributing the load between the network's available edge servers. By offloading requests from overloaded MEC servers to less crowded ones, the processing capacity and computational latency can be improved across the whole network. Finally, since the edge servers have reduced processing capacities compared to cloud servers, edge servers can also decide to offload time-sensitive requests to be executed at the cloud server.

DRL has been applied to task offloading problems. For instance, a DDPG-based task offloading algorithm shows better performance improvement compared to traditional offloading strategies (Chen et al., 2021). The joint binary task offloading and resource allocation is also studied where a parameterized policy is learned to map channel gains to offloading decisions (Huang et al., 2020). Partial task offloading has also been addressed using DRL. For instance, a cooperative multi-agent DRL framework was proposed where one agent selects the target server and the second decides the amount of data to be transferred (Lu et al., 2020). We refer the interested reader to Shakarami et al. (2020) for a comprehensive

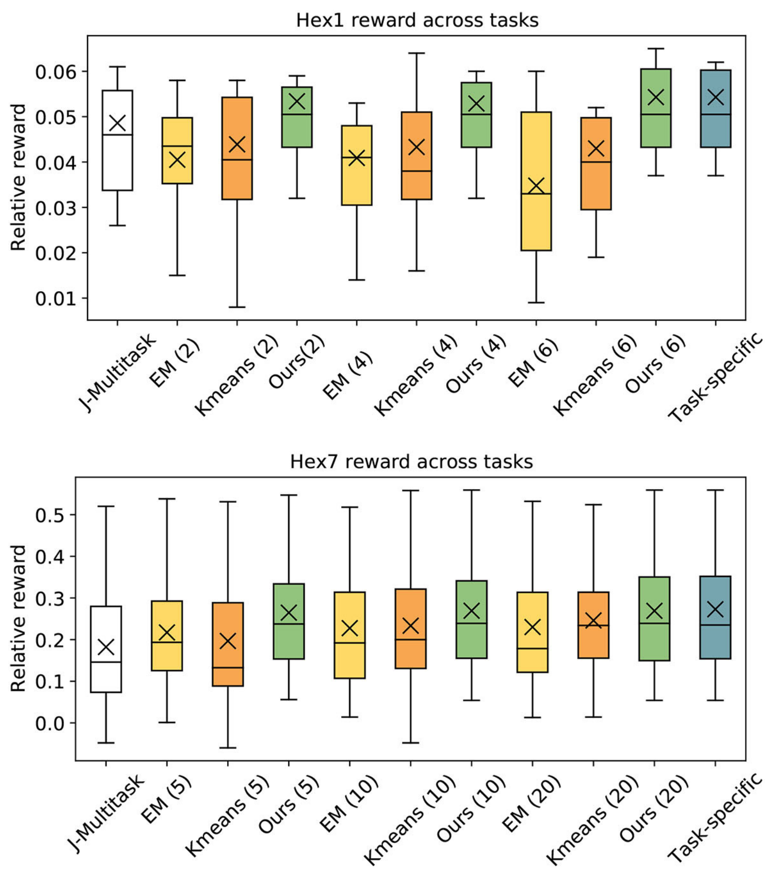

FIGURE 3
Reward across tasks achieved by each method relative to using fixed control parameters. Each box and whisker indicate quartiles, with the mean indicated by x and median indicated by a line in the box. Each clustering method is run with different numbers of clusters, shown in parentheses. The proposed method closely matches the task-specific models and achieves a higher mean than the other methods.

TABLE 2 Performance metrics (KPIs) achieved by all methods on Hex1 (left) and Hex7 (right) scenarios, relative to using fixed control parameters.

Method	Tput_{min}	Tput_{sd}	Tput_{avg}	$\text{Tput}_{<\chi}$	Method	Tput_{min}	Tput_{sd}	Tput_{avg}	$\text{Tput}_{<\chi}$
J-Multitask	+14%	−10%	+9%	−20%	J-Multitask	+23%	−22%	+12%	−12%
EM (2)	+12%	−7%	+7%	**−21%**	EM (5)	+26%	−25%	+10%	−20%
Kmeans (2)	+13%	−9%	+8%	−20%	Kmeans (5)	+24%	−24%	+10%	−16%
Ours (2)	**+15%**	**−11%**	**+9%**	**−21%**	Ours (5)	**+30%**	**−27%**	**+11%**	**−34%**
EM (4)	+12%	−9%	+8%	−18%	EM (10)	+27%	−23%	+10%	−26%
Kmeans (4)	+12%	−8%	**+9%**	−18%	Kmeans (10)	+27%	−25%	**+11%**	−24%
Ours (4)	**+15%**	**−11%**	**+9%**	**−21%**	Ours (10)	**+31%**	**−27%**	**+11%**	**−34%**
EM (6)	+11%	−6%	+6%	−12%	EM (20)	+27%	−25%	+11%	−27%
Kmeans (6)	+13%	−8%	+8%	**−23%**	Kmeans (20)	+29%	−25%	+11%	−27%
Ours (6)	**+16%**	**−11%**	**+9%**	−21%	Ours (20)	**+31%**	**−27%**	**+12%**	**−33%**
Task-specific	+16%	−11%	+9%	−22%	Task-specific	+31%	−27%	+12%	−33%

Numbers in parentheses indicate the number of clusters used. We bold the best result achieved for each cluster number.

survey on machine learning based computation offloading in MEC environments.

Task offloading and load balancing between edge servers is also examined to minimize the computational costs for all users (Lyu et al., 2022). The joint optimization problem is decomposed into two subproblems: a task offloading control scheme and a server grouping subproblem. The latter is solved using a value-based DRL algorithm. In a similar vein, Gao and Li (2022) studied the joint optimization of task offloading and load balancing. In this work, the task offloading problem is solved using the DDPG algorithm and particle swarm optimization is adopted to balance the load between the edge servers. Furthermore, Li et al. (2020) introduces an adaptive approach for joint load balancing and task scheduling in mobile fog computing networks.

A DRL-based solution for load balancing between MEC edge servers and the cloud is proposed by Tahmasebi-Pouya et al. (2022) where each MEC server is considered a separate agent. A Q-learning-based approach is adopted in which the agent decided either to offload requests to neighboring MEC servers or the cloud server. This work presented a fully decentralized solution where each MEC server is an independent learner relying on its local information only without any explicit information exchange between the edge servers. More recently, load balancing between edge servers in communication, computation and caching enabled heterogeneous MEC networks is studied using MARL (Ma et al., 2022). The problem is formulated as a user association problem where a separate agent decides the association action for each user.

6.2. Load balancing for WiFi (802.11) networks

Load balancing for WiFi networks involves distributing load across multiple access points and across the available frequency bands. The original 802.11 wireless standards were designed for uncoordinated access points. This limits performance in terms of handover, a critical component of load balancing. The actual process of handover is described in Pack et al. (2007). See also Mishra et al. (2003). Unlike the situation found in cellular network load balancing, the load balancing problem in 802.11 networks is complicated by the original design of the 802.11 network structure and the heterogeneous nature of the wireless network infrastructure.

The basic process of load balancing in 802.11 networks is described in Yen and Chi (2009). Depending on the strategy, load balancing in 802.11 networks can either be mediated by either the access point to which the device is connected, possibly in concert with other network-based resources, or by the device itself. For wireless device-based load balancing approaches, the wireless devices typically apply a rule-based approach to trigger handover to select a new access point independently. Even if such handovers are informed by centrally-managed network performance information, this approach typically does not result in effective network-wide load balancing (Yen and Chi, 2009).

An access point-based approach provides better centralized control of the network, possibly resulting in better load balancing. Individual AP's can control their load through adjusting their signal strength (coverage adjustment), rejecting new connections at heavily loaded access points (admission control), and disconnecting connected clients from their current access point (association management). The process of adjusting these processes is controlled by a rule-based system [as in Papanikos and Logothetis (2001) and Krishnan and Laxmi (2015)]. Szott et al. (2022) provides a broad survey of the use of machine learning (ML) approaches to the load balancing problem for WiFi networks. A range of different ML approaches exist in the literature including deep reinforcement learning-based approaches (e.g., Ali et al., 2019; Zhang et al., 2020).

7. Challenges, opportunities, and potential future directions

Reinforcement learning, especially deep reinforcement learning, has shown its effectiveness and superior performance in a range of different domains including load balancing in different wireless communication domains. In contrast to classical rule-based solutions, RL-based solutions can adjust the agents' behaviors dynamically in reaction to different system states and thus adapt quickly to optimal solutions. Given the current and anticipated future demand on the wireless network infrastructure, the previous rule-based approach to load balancing is no longer sufficient. RL-based load balancing appears to be a likely successor to previous approaches and play a more important role on optimizing network performance.

In spite of these advantages, there are still several challenges that hinder the applicability of reinforcement learning based LB solutions for real-world applications such as data efficiency, the lack of suitable simulator for simulation-based training and evaluation, safety, and explainability. Here, we briefly discuss these main challenges, offer some potential solutions and suggest potential directions for future research.

7.1. Data efficiency

Most deep reinforcement learning algorithms require a large number of interactions with the environment, i.e., more than one million interactions are often required to learn a reliable control policy. In the real world, such a large number of interactions, even if possible, will require a long training time. Meanwhile, real-time interactions with the networks can also introduce safety concerns. Data efficiency is one of the main challenges of bringing the RL-based solutions for real-world problems. There are a number of different types of methods that can be used to help improve data efficiency via using a certain type of prior, including transfer learning (Zhuang et al., 2020), meta learning (Finn et al., 2017), and data augmentation (Laskin et al., 2020). These and other approaches aim to reuse learned models, representations, and even the source samples to help improve the learning efficiency. Furthermore, it has also been shown that batch reinforcement learning (Wu et al., 2018; Fu et al., 2022a) and model based reinforcement learning (Huang et al., 2021) can also help future improve the data efficiency.

7.2. Safety

A critical concern related to deploying RL-based solutions for real-world applications relates to safety. Actions suggested by RL agents may bring an operating real-world system into some undesirable or even dangerous state. There two main aspects to such safety concerns. The first involves exploration in the model training phase. In order to learn a high-performance control policy, the RL agent needs to explore the environment, typically by taking random exploratory actions which can be dangerous in the real world. The other safety concern relates to distribution drift when operating. Though a trained RL agent may perform very well for a the task for which it was trained, if the properties of the task change, the performance of the RL agent may drop significantly. A number of potential solutions have been proposed to address safety concerns i.e., reward engineering (Dai et al., 2019), constrained optimization (Kamri et al., 2021), integrating with prediction (Krasowski et al., 2020). However, how to design a RL-based controller to efficiently and quickly adapt to distribution drift and manage a good trade off between safety and efficiency is still very challenging.

7.3. Simulation

Learning a reliable control policy requires the agent to interact with either the real environment or a simulator that can mimic the real world to a sufficiently high level of fidelity that results using the simulator apply in the real world (Mashaly, 2021; Lv et al., 2022). High simulator fidelity is typically associated with high computational cost and it can be extremely difficult to tune and validate the simulator. Unfortunately, today's simulators are either too slow or insufficiently precise. If the simulation is too slow, the RL model training process may take weeks or even months. On the other hand, if the simulation is far from the real world, the RL agent trained on the simulator may not perform well in the real world due to the well-known concerns, Sim2Real gap. The development of high performance, well-validated simulators is a key direction for future research. Furthermore developing reinforcement learning algorithms that can achieve better generalization will also be of critical importance.

7.4. Explainability

Another main obstacle of bringing RL based solutions to the real world is the lack of good explainabilities. Compared with the rule-based solutions, RL based solutions, especially for deep RL base solutions are mostly viewed as black boxes for the senior managers of telecom companies. To enable RL-based LB solutions easily adopted in real world, we need make the training process and the behavior logic easily to understand for the managers and domain experts. There are several recent works aiming to address the explainability concerns for deep reinforcement learning. However, for communication load balancing, this concern is not yet well-studied. As a summary, reinforcement learning based methods have already showcased impressive performance for communication load balancing. Meanwhile, there are still several aspects are still not well-studied.

Author contributions

DW, JL, AF, YX, and MJ wrote the main parts of this survey. SJ, XL, and GD participated the discussions and helped revised the submission.

Conflict of interest

SJ was employed by Samsung Electronics.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Åström, K. J. (1965). Optimal control of markov processes with incomplete state information. *J. Math. Anal. Appl.* 10, 174–205.

3GPP (2011). *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol Specification (Release 8)*. Technical report, 3GPP.

Aboelwafa, M., Alsuhli, G., Banawan, K., and Seddik, K. G. (2022). "Self-optimization of cellular networks using deep reinforcement learning with hybrid action space," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, NV), 223–229.

Ackermann, J., Richter, O., and Wattenhofer, R. (2021). "Unsupervised task clustering for multi-task reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Bilbao: Springer), 222–237.

Addali, K. M., Melhem, S. Y. B., Khamayseh, Y., Zhang, Z., and Kadoch, M. (2019). Dynamic mobility load balancing for 5G small-cell networks based on utility functions. *IEEE Access* 7, 126998–127011. doi: 10.1109/ACCESS.2019.2939936

Afroz, F., Subramanian, R., Heidary, R., Sandrasegaran, K., and Ahmed, S. (2015). SINR, RSRP, RSSI and RSRQ measurements in long term evolution networks. *Int. J. Wireless Mobile Netw.* 7, 113–123. doi: 10.5121/ijwmn.2015.7409

Ali, R., Shahin, N., Zikria, Y. B., Kim, B.-S., and Kim, S. W. (2019). Deep reinforcement learning paradigm for performance optimization of channel observation-based Mac protocols in dense WLANs. *IEEE Access* 77, 3500–3511. doi: 10.1109/ACCESS.2018.2886216

Alsuhli, G., Banawan, K., Seddik, K., and Elezabi, A. (2021a). "Optimized power and cell individual offset for cellular load balancing via reinforcement learning," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)* (Las Vegas, NV), 1–7.

Alsuhli, G., Ismail, H. A., Alansary, K., Rumman, M., Mohamed, M., and Seddik, K. G. (2021b). "Deep reinforcement learning-based cio and energy control for lte mobility load balancing," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, NV), 1–6.

Asghari, M. Z., Ozturk, M., and Hämäläinen, J. (2021). "Reinforcement learning based mobility load balancing with the cell individual offset," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)* (Helsinki), 1–5.

Attiah, K., Banawan, K., Gaber, A., Elezabi, A., Seddik, K., Gadallah, Y., et al. (2020). "Load balancing in cellular networks: A reinforcement learning approach," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, NV), 1–6.

Chen, M., Wang, T., Zhang, S., and Liu, A. (2021). Deep reinforcement learning for computation offloading in mobile edge computing environment. *Comput. Commun.* 175, 1–12. doi: 10.1016/j.comcom.2021.04.028

Dai, C., Xiao, L., Wan, X., and Chen, Y. (2019). "Reinforcement learning with safe exploration for network security," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Brighton: IEEE), 3057–3061.

Ericsson (2022). *Ericsson Mobility Report Data and Forecasts*. Available online at: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts.

Feriani, A., Wu, D., Xu, Y. T., Li, J., Jang, S., Hossain, E., et al. (2022). Multiobjective load balancing for multiband downlink cellular networks: a meta-reinforcement learning approach. *IEEE J. Select. Areas Commun.* 40, 2614–2629. doi: 10.1109/JSAC.2022.3191114

Finn, C., Abbeel, P., and Levine, S. (2017). "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning* (Sydney: PMLR), 1126–1135.

Fu, Y., Wu, D., and Boulet, B. (2022a). "A closer look at offline Rl agents," in *Thirty-Sixth Conference on Neural Information Processing Systems (NeurIPS 2022)*, eds S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (New Orleans, LO), 8591–8604.

Fu, Y., Wu, D., and Boulet, B. (2022b). "Reinforcement learning based dynamic model combination for time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 6639–6647.

Fujimoto, S., Hoof, H., and Meger, D. (2018). "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning* (Stockholm: PMLR), 1587–1596.

Gao, Y., and Li, Z. (2022). "Load balancing aware task offloading in mobile edge computing," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (Hangzhou), 1209–1214.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*. doi: 10.48550/arXiv.1801.01290

Hendrawan, H., Zain, A., and Lestari, S. (2019). Performance evaluation of A2-A4-RSRQ and A3-RSRP handover algorithms in LTE network. *Jurnal Elektronika dan Telekomunikasi* 19, 64–74. doi: 10.14203/jet.v19.64-74

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. doi: 10.48550/arXiv.1503.02531

Huang, L., Bi, S., and Zhang, Y.-J. A. (2020). Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Trans. Mobile Comput.* 19, 2581–2593. doi: 10.1109/TMC.2019.2928811

Huang, X., Wu, D., Jenkin, M., and Boulet, B. (2021). Modellight: model-based meta-reinforcement learning for traffic signal control. *arXiv preprint arXiv:2111.08067*. doi: 10.48550/arXiv.2111.08067

Jorguseski, L., Pais, A., Gunnarsson, F., Centonza, A., and Willcock, C. (2014). Self-organizing networks in 3GPP: standardization and future trends. *IEEE Commun. Mag.* 52, 28–34.

Kamri, A. Y., Quang, P. T. A., Huin, N., and Leguay, J. (2021). "Constrained policy optimization for load balancing," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)* (Milan: IEEE), 1–6.

Kang, J., Chen, X., Wu, D., Xu, Y. T., Liu, X., Dudek, G., et al. (2021). "Hierarchical policy learning for hybrid communication load balancing," in *ICC 2021-IEEE International Conference on Communications* (Montreal, QC: IEEE), 1–6.

Krasowski, H., Wang, X., and Althoff, M. (2020). "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *2020 IEEE 23rd international conference on Intelligent Transportation Systems (ITSC)* (Rhodes: IEEE), 1–7.

Krishnan, R., and Laxmi, V. (2015). "IEEE 802.11 WLAN load balancing for network enhancement," in *3rd Int. Conf. on Recent Trends in Computing* (Ghaziabad).

Kwan, R., Arnott, R., Paterson, R., Trivisonno, R., and Kubota, M. (2010). "On mobility load balancing for LTE systems," in *2010 IEEE 72nd Vehicular Technology Conference-Fall* (Ottawa: IEEE), 1–5.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. (2020). "Reinforcement learning with augmented data," *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, NeurIPS 2020, eds H. Larochelle, M. Ranzato, R. Hadsell, M. -F. Balcan, and H.-T. Lin. p. 19884–19895.

Li, J., Wu, D., Xu, Y. T., Li, T., Jang, S., Liu, X., et al. (2022). "Traffic scenario clustering and load balancing with distilled reinforcement learning policies," in *ICC 2022-IEEE International Conference on Communications* (Seoul: IEEE), 1536–1541.

Li, X., Qin, Y., Zhou, H., Chen, D., Yang, S., and Zhang, Z. (2020). An intelligent adaptive algorithm for servers balancing and tasks scheduling over mobile fog computing networks. *Wireless Commun. Mobile Comput.* 2020, 1–16. doi: 10.1155/2020/8863865

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*. doi: 10.48550/arXiv.1509.02971

Lu, H., Gu, C., Luo, F., Ding, W., and Liu, X. (2020). Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Gen. Comput. Syst.* 102, 847–861. doi: 10.1016/j.future.2019.07.019

Lv, Z., Chen, D., Feng, H., Wei, W., and Lv, H. (2022). Artificial intelligence in underwater digital twins sensor networks. *ACM Trans. Sensor Netw.* 18, 1–27. doi: 10.1145/3519301

Lyu, F., Dong, Z., Wu, H., Duan, S., Wu, F., Zhang, Y., et al. (2022). "Mobility-aware computation offloading with adaptive load balancing in small-cell MEC," in *ICC 2022 - IEEE International Conference on Communications* (Seoul), 4330–4335.

Ma, M., Wu, D., Xu, Y. T., Li, J., Jang, S., Liu, X., et al. (2022). "Coordinated load balancing in mobile edge computing network: a multi-agent DRL approach," in *ICC 2022 - IEEE International Conference on Communications* (Seoul), 619–624.

Mashaly, M. (2021). Connecting the twins: a review on digital twin technology & its networking requirements. *Proc. Comput. Sci.* 184, 299–305. doi: 10.1016/j.procs.2021.03.039

Mishra, A., Shin, M., and Arbaugh, W. (2003). An empirical analysis of the IEEE 802.11 MAC layer handoff process. *ACM SIGCOMM Comp. Commun. Rev.* 33, 93–102. doi: 10.1145/956981.956990

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning* (New York, NY), 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*. doi: 10.48550/arXiv.1312.5602

Munoz, P., Barco, R., de la Bandera, I., Toril, M., and Luna-Ramirez, S. (2011). "Optimization of a fuzzy logic controller for handover-based load balancing," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)* (Budapest: IEEE), 1–5.

Munoz, P., Barco, R., Ruiz-Aviles, J. M., de la Bandera, I., and Aguilar, A. (2013). Fuzzy rule-based reinforcement learning for load balancing techniques in enterprise LTE femtocells. *IEEE Trans. Vehicular Technol.* 62, 1962–1973. doi: 10.1109/TVT.2012.2234156

Musleh, S., Ismail, M., and Nordin, R. (2017). Load balancing models based on reinforcement learning for self-optimized Macro-Femto LTE-advanced heterogeneous network. *J. Telecommun. Electron. Comput. Eng.* 9, 47–54.

Mwanje, S. S., and Mitschele-Thiel, A. (2013). "A Q-learning strategy for LTE mobility load balancing," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (London), 2154–2158.

Mwanje, S. S., Schmelz, L. C., and Mitschele-Thiel, A. (2016). Cognitive cellular networks: a Q-learning framework for self-organizing networks. *IEEE Trans. Netw. Serv. Manage.* 13, 85–98. doi: 10.1109/TNSM.2016.2522080

Nian, R., Liu, J., and Huang, B. (2020). A review on reinforcement learning: introduction and applications in industrial process control. *Comput. Chem. Eng.* 139, 106886. doi: 10.1016/j.compchemeng.2020.106886

Pack, S., Choi, J., Kwon, T., and Choi, Y. (2007). Fast-handoff support in IEEE 802.11 wireless networks. *IEEE Commun. Surv. Tutorials* 9, 2–12. doi: 10.1109/COMST.2007.358968

Papanikos, I., and Logothetis, M. (2001). "A study on dynamic load balance for IEEE 802.11b wireless LAN," in *Proc. 8th Intl Conf. Advances in Communication and Control* (Crete), 83–89.

Pinto, L., and Gupta, A. (2017). "Learning to push by grasping: Using multiple tasks for effective learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (Marina Bay Sands: IEEE), 2161–2168.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons.

Puttonen, J., Kolehmainen, N., Henttonen, T., and Kaikkonen, J. (2009). "On idle mode mobility state detection in evolved utran," in *2009 Sixth International Conference on Information Technology: New Generations* (Las Vegas, NV: IEEE), 1195–1200.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*. doi: 10.48550/arXiv.1511.05952

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). "Trust region policy optimization," in *International Conference on Machine Learning* (Lille), 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. doi: 10.48550/arXiv.1707.06347

Shakarami, A., Ghobaei-Arani, M., and Shahidinejad, A. (2020). A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective. *Comput. Netw.* 182, 107496. doi: 10.1016/j.comnet.2020.107496

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning* (Bejing), 387–395.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning and Machine Learning: An Introduction, 2nd Edn.* MIT Press. Available online at: https://www.worldcat.org/oclc/37293240

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, 12, eds S. A. Solla, T. K. Leen, and K. -R. Muller (Denver, CO), 1057–1063.

Szott, S., Kosek-Szott, K., Gawlowicz, P., Gómez, J. T., Bellalta, B., Zubow, A., et al. (2022). WiFi meets ML: a survey ion improving IEEE 802.11

performance with machine learning. *IEEE Commun. Surv. Tutorials* 24, 1843–1893. doi: 10.1109/COMST.2022.3179242

Tahmasebi-Pouya, N., Sarram, M.-A., and Mostafavi, S.-A. (2022). "Load balancing in mobile edge computing: a reinforcement learning approach," in *2022 Sixth International Conference on Smart Cities, Internet of Things and Applications (SCIoT)* (Mashhad), 1–6.

Van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence* (Phoenix), 30.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning* (New York, NY), 1995–2003.

Watkins, C. J., and Dayan, P. (1992). Q-learning. *Mach. Learn.* 8, 279–292.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 229–256.

Wu, D. (2018). *Machine Earning Algorithms and Applications for Sustainable Smart Grid*. Montreal: McGill University.

Wu, D., Kang, J., Xu, Y. T., Li, H., Li, J., Chen, X., et al. (2021). "Load balancing for communication networks via data-efficient deep reinforcement learning," in *2021 IEEE Global Communications Conference (GLOBECOM)* (Madrid), 1–7.

Wu, D., Rabusseau, G., François-lavet, V., Precup, D., and Boulet, B. (2018). "Optimizing home energy management and electric vehicle charging with reinforcement learning," in *2018 ICML Workshop on Adapative Learning Agent* (Stockholm).

Xu, Y., Xu, W., Wang, Z., Lin, J., and Cui, S. (2019a). "Deep reinforcement learning based mobility load balancing under multiple behavior policies," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)* (Shanghai: IEEE), 1–6.

Xu, Y., Xu, W., Wang, Z., Lin, J., and Cui, S. (2019b). Load balancing for ultradense networks: a deep reinforcement learning-based approach. *IEEE Internet Things J.* 6, 9399–9412. doi: 10.1109/JIOT.2019.29 35010

Yen, L.-H., and Chi, K.-H. (2009). Load balancing in IEEE 802.11 networks. *IEEE Internet Comput*. 13, 56–64. doi: 10.1109/MIC.2009.11

Zhang, L., Yin, H., Zhou, Z., Roy, S., and Sun, Y. (2020). "Enhancing wifi multiple access performance with federated deep reinforcement learning," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)* (Victoria, BC), 1–6.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., et al. (2020). A comprehensive survey on transfer learning. *Proc. IEEE* 109, 43–76. doi: 10.1109/JPROC.2020. 3004555

Nomenclature

TABLE 3 Summary of key notations and symbols.

Load balancing	
X_i	Signal quality of the i-th cell (e.g., RSRP)
C	Number of cells in the network
$T_{\mathrm{CIO}_{i,j}}$	Cell Individual Offset (CIO) between the cells i and j
$Hyst$	Hysteresis parameter
$T_{\mathrm{A4}_i}, T_{\mathrm{A2}_i}$	Controllable threshold of the i-th for the A4 and A2 events, respectively
$T_{\mathrm{A5}_{i,j}}, T'_{\mathrm{A5}_{i,j}}$	Pairwise thresholds of the A5 event between the cells i and j
$T_{\mathrm{CRS}_{i,j}}$	Pairwise Cell re-selection threshold between the cells i and j
ρ	Physical resource block utilization percentage
$f_{u,i}$	Number of packets transmitted to the UE u from the cell i
$d_{u,i}$	Delay experienced by the UE u when served by the cell i
$\mathrm{Tput}_{u,i}$	Throughput of the UE u when served by the cell i
U, U_i	The set of connected UEs to the network, the subset of UEs connected to a cell i
Reinforcement learning	
$\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$	State, action spaces, transition probability function, scalar reward function
γ, κ_0	Discount factor, initial state distribution
$\mathcal{O}, \zeta, b$	Observation space, the probability distribution over observations given a state and an action and the belief state, respectively
$\pi, \pi^*, \Pi, \pi_\phi$	A policy, the optimal policy, the set of all possible policies, a parameterized policy with parameters ϕ
$\mathcal{J}$	Expected cumulative discounted reward
V^π, Q^π, A^π	The value function, the state-action value function and the advantage function under a policy π
$Q^*, \hat{Q}, \bar{Q}, Q_\theta$	Optimal Q-function, estimated Q-function, target Q-function, parameterized Q-function with parameters θ
α	Learning rate
D	Replay buffer
H	Episode horizon or length of a trajectory

Here, we present a summary of the notations used in this paper, as shown in Table 3.