



Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

journal homepage: www.elsevier.com/locate/jpdc

Computation offloading in collaborative computing systems: A comprehensive survey on methods, challenges, and future directions

Jussi Kalliola^{a,*}, Jason J. Jung^b, David Hästbacka^a^a Tampere university, Faculty of Information Technology and Communication Sciences, Korkeakoulunkatu 1, Tampere, 33720, Finland^b Chung-Ang University, 84 Heukseok-Ro, Dongjak-Gu, Seoul, 06974, South Korea

ARTICLE INFO

Keywords:

Collaborative computing
Edge computing
Distributed systems
Computation offloading

ABSTRACT

The rapid increase in the number of connected devices has created a vast computational infrastructure. While intelligent applications (e.g., VR/AR, autonomous driving, and AI assistants) demand substantial processing power, they are typically executed only in short bursts. As a result, devices remain idle for significant periods, presenting an opportunity to utilize their unused computational resources. Computation offloading has emerged as a viable solution, where resource-constrained devices leverage edge or cloud infrastructure for intensive tasks. More recently, horizontal offloading has gained attention as a complementary approach, enabling devices within the same layer to collaborate and utilize idle resources more efficiently. In this survey, we examine current research on computation offloading in collaborative computing systems that span from horizontal offloading among end devices to more complex systems involving horizontal as well as vertical offloading to edge/cloud systems. Our examination focuses on architectures, collaborative characteristics, and offloading dynamics, such as task dependency, mobility, and multi-hop offloading. We begin by introducing edge computing paradigms and key concepts in computation offloading and collaborative computing. The offloading approaches are then classified into classical, heuristic, metaheuristic, and machine learning methods, further divided into centralized, decentralized, and distributed categories. These approaches are analyzed and compared based on collaborative system, architecture, offloading dynamics, and evaluation methodology. Finally, we have identified open challenges and emphasized the need to unify research across edge computing paradigms, alongside a focus on more complex collaborative systems that resemble real-world systems, and the development of standardized evaluation frameworks as key directions for future work.

1. Introduction

In recent years, the computational capabilities of battery-powered mobile devices have seen significant advancements. Enhancements in chip design, hardware accelerators, memory, and connectivity have transformed these devices into powerful, multipurpose tools suitable for a wide range of applications, including computer vision and natural language processing. These improvements are particularly critical in fields like robotics, Augmented/Virtual Reality (AR/VR), and self-driving vehicles, where devices must process and interpret data from dynamic environments through various sensors. However, processing and analyzing sensory data still demands high computational power, which remains a challenge for battery-powered devices. Moreover, intensive computational tasks consume significant energy, limiting the device's operational time.

A prominent solution to this issue is *Computation offloading*, where computationally intensive tasks are offloaded to more resource-rich de-

vices, such as those in the cloud. While cloud computing has been a widely adopted paradigm, it suffers from high latency [1]. This makes cloud computing a less favorable option for real-time applications, such as self-driving vehicles and robotics, as it may not meet critical requirements like reliability, and rapid responsiveness due to variance in wider networks [2]. To address this, the edge computing paradigm has emerged, placing servers closer to mobile devices to reduce latency. As a result, edge computing has gained increasing popularity in mobile and vehicular applications, including robotics and aerial vehicles [2–4].

In recent years, Device-to-Device (D2D) offloading has gained increasing attention as a strategy to enhance the utilization of idle devices, particularly due to advancements in computational capabilities and communication technologies of user devices. This is especially important for applications with bursty computational demands and low-latency requirements, such as AR/VR, Artificial Intelligence (AI) assistants, and self-driving systems, which often alternate between high processing loads and idle periods. D2D offloading leverages this idle

* Corresponding author.

E-mail addresses: jussi.kalliola@tuni.fi (J. Kalliola), j3jung@cau.ac.kr (J.J. Jung), david.hastbacka@tuni.fi (D. Hästbacka).<https://doi.org/10.1016/j.jpdc.2026.105284>

Received 22 April 2025; Received in revised form 17 October 2025; Accepted 2 May 2026

Available online 5 May 2026

0743-7315/© 2026 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

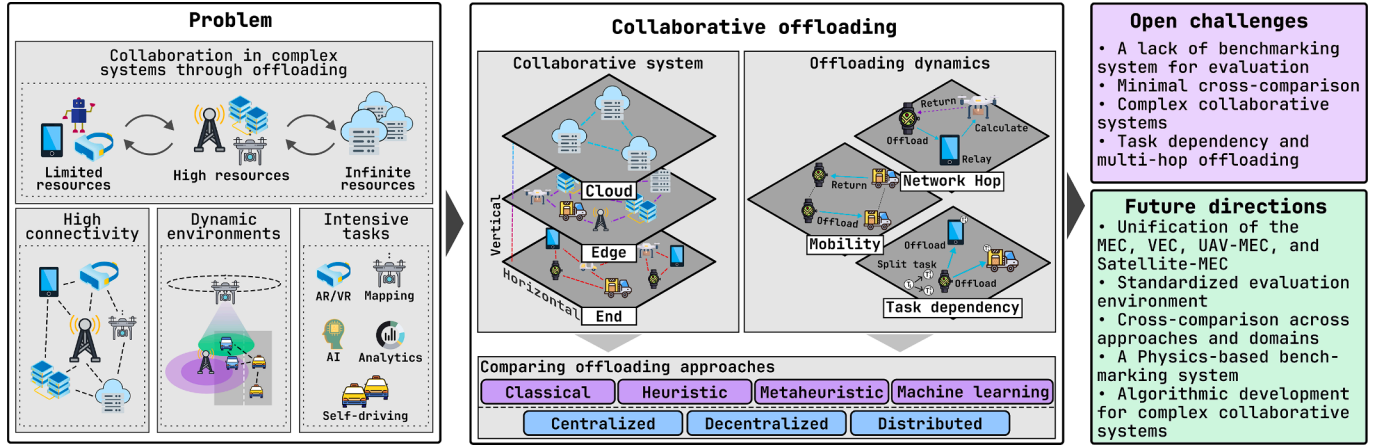


Fig. 1. Overview of this survey.

capacity by enabling devices with similar resources to collaborate and offload tasks within direct communication range. This approach can achieve lower latency; however, its performance is constrained by the computational capabilities of individual devices. The term *Collaborative Computing* refers to the utilization of communication technologies by devices to share resources or exchange information. In this survey, we define a collaborative system as one in which all devices within the network, including edge servers, the cloud, and other mobile devices, are capable of sharing information and resources.

1.1. Computation offloading

Computation offloading involves transferring a computational task from one device to another, which performs the task on behalf of the original device. Offloading is particularly useful when a device lacks the necessary resources, whether due to temporary limitations or permanent deficiencies, such as inadequate computational power, memory, or data availability. It also helps accelerate computation time and balance workloads across a network of devices, especially when a device needs to allocate resources for higher-priority tasks.

Offloading is typically framed as an optimization problem, aiming to optimize a specific objective, which is often application-dependent [5]. The problem can be formulated as single or multi-objective [6]. Common objectives include minimizing energy consumption, latency, or resource allocation [5]. Other novel objectives have been proposed, such as social trust [7] or satisfaction [8]. A variety of methods have been proposed to solve the optimization problem, ranging from classical approaches to machine learning techniques. Given that the original problem is often NP-hard, decomposition or relaxation techniques are frequently used [5] to make the problem more tractable. Additionally, strategies for solving the offloading problem and its subproblems often combine multiple techniques from different method categories, such as classical and heuristic approaches [7,9], as well as metaheuristic and machine learning methods [10]. In this article, we categorize the task offloading methods into classical, heuristic, metaheuristic, and machine learning approaches, which are further subdivided into centralized, decentralized, and distributed approaches. Centralized methods are employed to solve global optimization problems by leveraging a global view of the environment. In contrast, distributed methods aggregate data from individual agents to collectively address a global optimization problem. Decentralized methods, on the other hand, involve each agent solving an optimization problem based solely on local information.

1.2. Collaborative computing

In collaborative computing systems, a device shares information, resources, or seeks assistance from other devices to perform a task.

Typically, the requesting device is a mobile device, vehicle, or user equipment, while the assisting device may be another mobile device, a resource-rich edge server, or the cloud. A common architecture divides devices into three layers based on capabilities, and resources. The first layer, consisting of user equipment, vehicles, and other mobile devices, is considered the end layer, located at the edge of the network. The next layer, often referred to as the edge, includes devices with more computational resources. The edge layer aims to bring cloud capabilities closer to the end devices, reducing latency compared to traditional cloud computing. Specialized subsets of edge computing are Mobile Edge Computing (MEC) [11] designed for mobile networks, Vehicular Edge Computing (VEC) [12] for vehicular networks, Unmanned Aerial Vehicle (UAV) assisted MEC (UAV-MEC) [13] integrates edge capabilities to UAV, and Satellite-assisted MEC [14] that integrates edge server to a satellite. Edge and fog computing are often used interchangeably due to their shared characteristics, such as localized storage and computing near end devices. However, fog computing is generally considered more scalable and positioned even closer to end devices [15]. The last layer, cloud, is often considered to have virtually infinite resources [16]. Additionally, cloud and edge computing have been integrated with cognitive computing, enabling systems to learn from their environment and dynamically adapt, achieving human-like cognitive intelligence [17,18].

Collaboration can occur either vertically or horizontally. In vertical collaboration, also known as inter-layer, a device seeks assistance from a computing layer with more resources, such as the edge or cloud. This approach is particularly effective when a mobile device lacks sufficient capabilities. In horizontal collaboration, also known as intra-layer, a device seeks assistance from other devices within the same layer. This approach can achieve the lowest latency, particularly at the end layer, where devices are often in close proximity to one another. One of the primary challenges in collaborative computing is limited connectivity due to the mobility of devices [19]. Device movement can alter the network topology, introduce additional latency, or even push devices out of communication range [19].

1.3. Contribution and organization

This article provides a comprehensive survey of 71 studies on task offloading in collaborative systems, including MEC, VEC, and UAV-MEC. The overview of the article can be seen in the Fig. 1. It examines computation offloading approaches within various collaborative environments, with a particular focus on intra- and inter-layer offloading dynamics. These dynamics offer valuable insights into the underlying principles of methods applied in complex systems. Our key contributions are summarized as follows:

Table 1
Comparison of related surveys.

	Year	Collaborative task offloading			Network			Focus on offloading	Focus on collaboration	Main Perspective
		D2D	Edge	Cloud	MEC	VEC	Aerial			
[20]	2018	✓			✓				✓	Multi-device collaboration in MEC
[21]	2019		✓	✓	✓	✓		✓		Various aspects (energy, QoS, QoE)
[22]	2020		✓	✓	✓			✓		Various offloading modeling methods
[23]	2020		✓		✓			✓		Game-theoretic approaches to offloading
[24]	2020		✓	✓	✓	✓			✓	Edge-cloud task offloading taxonomy
[19]	2021	✓	✓	✓	✓			✓		Mobility-aware MEC offloading
[5]	2022	✓	✓	✓	✓	✓	✓	✓		Most common offloading issues
[6]	2022		✓	✓	✓	✓		✓		Optimization methods for offloading
[4]	2022		✓	✓	✓	✓	✓	✓		Offloading for ground and aerial robots
[3]	2022	✓	✓	✓		✓		✓	✓	Vehicular task offloading taxonomy
[13]	2022		✓				✓	✓		UAV-MEC offloading
[25]	2022		✓	✓	✓	✓		✓		MEC offloading: Issues in objectives
[26]	2022		✓		✓			✓		Joint WPT-MEC energy solutions
[27]	2022		✓	✓	✓			✓		Focus on ML, EA, SIA methods
[28]	2022	✓	✓	✓	✓	✓	✓	✓		RL methods for offloading
[29]	2024	✓	✓	✓	✓	✓	✓	✓		DRL for offloading
[30]	2024		✓	✓	✓			✓	✓	MADRL and DRL for offloading
[31]	2024		✓	✓	✓	✓	✓	✓		Focus on task types for offloading
[32]	2024		✓	✓	✓	✓	✓	✓		Offloading strategies in MEC
[33]	2025		✓	✓	✓	✓	✓	✓		Game-theoretic approaches to offloading
[34]	2025		✓	✓			✓	✓		Offloading from low-altitude to satellites
[35]	2025	✓	✓	✓	✓				✓	Federated and distributed offloading in MEC
Our	2025	✓	✓	✓	✓	✓	✓	✓	✓	Offloading strategies in collaborative systems

1. We present collaborative system architectures, system models, and offloading approaches, providing a detailed discussion of their key components.
2. We offer a thorough survey of both fundamental and recent advancements in computation offloading in collaborative systems that span from D2D offloading to more complex offloading schemes, emphasizing intra- and inter-layer dynamics such as mobility, multi-hop offloading, and task dependency. We categorize the offloading approaches into four groups and further classify them based on their design into centralized, decentralized, and distributed.
3. We highlight the open challenges in the field and propose potential research areas. It includes discussions on the contributions of algorithms, the development of evaluation environments, the unification of offloading in edge computing research, and the exploration of complex collaborative systems.

The article is organized as follows: [Section 2](#) covers the search methodology, and [Section 3](#) related work. [Section 4](#) provides the preliminaries of collaborative computing, system modeling, and computation offloading. [Section 5](#) reviews the literature, categorizing and comparing articles based on offloading approaches. [Section 6](#) discusses open challenges and proposes future research directions, followed by the conclusion in [Section 7](#).

2. Search methodology

The primary articles were searched from well-known databases including IEEE Xplore, Springer, ScienceDirect, ACM, and Scopus, without applying search restrictions such as publication year limits. The search query used across all libraries was: (“*edge computing*”) AND (*device-to-device OR vehicle-to-vehicle*) AND (*collaborative OR collaboration*) AND (*task OR computation*) AND (*allocation OR orchestration OR offloading*). In ScienceDirect, the query was split into two parts due to boolean connector limitations.

Rigorous screening was conducted to include relevant articles. We performed “Title and Abstract” and “Full Text” screenings to exclude irrelevant papers. Final exclusions were made during the analysis phase. Papers were included if they were peer-reviewed, written in English, and represented final publications (excluding pre-prints, theses, and patents). Only papers proposing an offloading method that involved

at least local computing and horizontal offloading among end devices (D2D) were selected, excluding papers that focused on other problems like resource allocation or solely vertical offloading (edge or cloud computing). In total, **71** articles between 2017 and 2025 were selected for analysis from an initial pool of **358**.

3. Related work

The reviewed surveys collectively address various aspects of computation offloading in MEC and related paradigms. Jiang et al. [21] focus on computation offloading using heuristic and game theory based approaches in edge computing, emphasizing energy consumption, Quality of Service (QoS) guarantee, and Quality of Experience (QoE) enhancement, while Zhang et al. [20] explore collaborative multi-device computation. Shakarami et al. [23] examine game theory-based offloading, categorizing, and comparing approaches in terms of utilized techniques, and case studies. In 2024, Pirhosseini et al. [33] revisited game-theory based offloading and presented systematic review with focus on offloading in edge-fog-cloud systems. Wang et al. [24] introduced a taxonomy for edge-cloud offloading and analyzed offloading methods based on task type, offloading schemes, optimization objectives, multi-hop cooperation, and mobility considerations. Lin et al. [22] focus on offloading modeling, including heuristic, machine learning, and game theory approaches, while Zaman et al. [19] emphasize mobility-aware offloading techniques. Sadatdiynov et al. [6] review six optimization approaches, advocating for real-world evaluations, and three-layer offloading scenarios. Mohammad et al. [5] provide a comprehensive mixed-method survey, identifying optimization approaches, offloading-related issues, and bibliometric trends.

In domain-specific contexts, Asiful Huda et al. [13] explore UAV-MEC offloading, which was revisited in 2025 by Ahmadun et al. [34], where they examined offloading approaches from low-altitude aerial vehicles up to satellites. Ahmed et al. [3] reviewed vehicular offloading under a communication perspective, with classification of methods under Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Everything (V2X) paradigms. Chaari et al. [4] focus on robotic offloading challenges like connectivity loss and fault tolerance, and Mustafa et al. [26] integrate Wireless Power Transfer (WPT) and MEC, emphasizing antenna design and simulation tools. Feng et al. [25] and

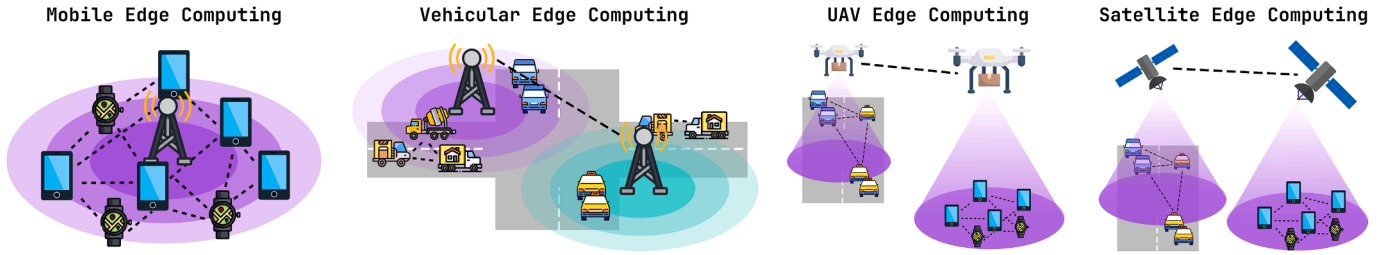


Fig. 2. Edge computing paradigms.

Jin et al. [27] focus on computation offloading in MEC covering wide variety of approaches, such as Machine Learning (ML), Evolutionary Algorithms (EA), and Swarm Intelligent Algorithms (SIA). Peng et al. [29] and Wei et al. [28] highlight Reinforcement Learning (RL) and Deep RL (DRL) methods. In 2024, Gu et al. [30] extended DRL approaches to Multi-agent RL (MARL) to optimize offloading, resource allocation, caching, and mobility management. Dong et al. [32] and Zhang et al. [31] provides the latest reviews of task offloading in MEC, where [31] focus on task type classification, and [32] categorizes methods based on objectives. Lastly, Pournazari et al. [35] in 2025 examined federated offloading approaches within Internet of Things (IoT) edge-to-cloud continuum. It categorized solutions into centralized and decentralized methods based on horizontal and vertical offloading. Their work is closely related to ours, yet our study extend it by comparing methods across computing paradigms, offloading dynamics, and by introducing a broader categorization into distributed models.

Collectively, these studies offer a comprehensive review of offloading approaches in various end-edge-cloud computing paradigms, covering domain-specific contexts, diverse objectives, and key challenges for advancing offloading and edge computing research (Table 1). However, previous works lack a comprehensive survey that addresses task offloading across various collaborative systems, encompassing all edge computing paradigms, computing modes, and various offloading dynamics while providing a multi-variable comparison of approaches based on architecture, edge computing paradigm, evaluation environment, and offloading dynamics.

4. Computation offloading in collaborative computing systems: Background

This section provides the preliminaries of collaborative computing and computation offloading. It introduces the fundamentals of collaborative computing and relevant computing paradigms, such as edge and cloud, followed by a detailed description of common system models, and computation offloading approaches.

4.1. Collaborative computing systems

Collaborative computing involves multiple devices or systems working together to solve computational problems, share resources or information, or coordinate actions toward a common goal. It integrates edge computing, cloud computing, and Peer-to-Peer (P2P) networks into a unified paradigm [5].

4.1.1. Computing architectures

Computing architectures are typically divided into two-tier and three-tier systems [22]. Two-tier architectures are common and consist of edge/fog and cloud computing [15,22]. Although cloud computing has been a long-standing solution, it suffers from high latency [1] which makes it less suitable for real-time applications [2]. Edge computing addresses this issue by deploying servers closer to the data source, thus reducing latency [36]. Three-tier architectures utilize all layers of the computing continuum [16,37], comprising end, edge, and cloud layers

[22]. These layers are typically defined by their computational capabilities and proximity to the network edge. End devices, including IoT devices, user equipment, robots, and vehicles, reside at the outermost layer and are responsible for generating data and tasks. Edge servers and devices constitute the middle layer, offering more resources than end devices and lower latency compared to the cloud. Cloud servers, with the highest resource capacity, make up the innermost layer. Additionally, integrating cognitive computing into edge or cloud paradigms allows systems to perceive their environment, learn from it, and dynamically adjust workloads [17,18].

Specialized edge computing subsets have been developed for various networks. MEC, for example, is designed specifically for cellular networks, where MEC servers are often deployed in mobile base stations [11]. VEC, an extension of MEC, deploys servers at Roadside Units (RSUs) to process data generated by vehicles [12]. UAV and Satellite-assisted MEC have gained attention in recent years, with MEC servers deployed on UAVs [13] or satellites [14], making this approach effective for applications requiring mobile edge servers. A visualization of the edge computing paradigms is presented in Fig. 2.

Across all computing paradigms, common requirements include minimizing latency (both transmission and computation), supporting device mobility, managing distributed resources, ensuring scalability, maximizing QoS, and enhancing energy efficiency [5].

4.1.2. Vertical collaboration

Vertical collaboration involves inter-layer devices working together to solve computational problems or share resources [5]. It shares similarities with edge or cloud computing, as resource-rich servers in these layers assist mobile devices. It is a prominent solution to accelerate computation time when a mobile device lacks resources and computational power.

4.1.3. Horizontal collaboration

Horizontal collaboration involves intra-layer devices cooperating with each other [5]. This is often referred to as D2D or V2V offloading [38–40]. In this article, intra-layer communication and cooperation are termed P2P to include all edge computing networks. Horizontal collaboration typically takes place between end devices, although some studies extend it to the edge layer to balance workloads or relocate computation closer to the end device after a change in its location [41,42]. End devices share resources and assist each other when extremely low latency is needed, facilitated by P2P connectivity.

Collaboration in this context is managed through an offloading strategy, which determines which tasks to offload, to what device, and when. The strategy can be controlled by a centralized server or individual devices. A detailed description of offloading is provided in Section 4.3.

4.2. System modeling

In this section, we define common models for collaborative systems, focusing on the system, energy, communication, task, computation, and mobility models. These models are used to represent realistic aspects of the system.

4.2.1. System model

The system is described at a high level, including definitions of computing layers, computing units, and the time model. The system is modeled as a set of end devices, denoted as $U = \{u_1, u_2, \dots, u_n\}$, a set of edge servers, denoted as $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$, and a cloud server C [43,44]. Another system model divides U into task devices and idle devices [44–46], or treats all computing units as a single set [47]. The MEC model typically defines U as mobile devices or user equipment, with $\mathcal{E}$ deployed at base stations [48–50]. In extensions like VEC, U represents vehicles, and $\mathcal{E}$ is deployed at cell towers or RSUs [51–53]. In this article, we use the unified notations of U , $\mathcal{E}$, and C for end devices, edge servers, and the cloud server, respectively.

A discrete time-slotted model is commonly used to describe the operational timeline. The timeline is defined as $\tau = \{t|1, T\}$ where each time slot t in τ has a duration of Δ seconds [40,43,54,55]. Various actions can occur during each time t , such as the movement of end devices [56], or task generation [43,57]. Many studies consider the system to be quasi-static within each time slot [40,56]. In this context, a quasi-static scenario assumes stable channel states, device movement, or other dynamic processes [56], though these conditions may change between time slots [40,56]. The single-timescale model was extended to a dual-timescale model in [47], where larger time scales are used for caching decisions and smaller time scales are used for offloading decisions.

4.2.2. Communication model

The communication model describes the interaction and data exchange between devices, often including data transmission rate, protocol, and transmission time [6]. P2P and Peer-to-Infrastructure (P2I) communications are typically modeled as a wireless link, while edge-to-cloud communication is treated as a wired link. Various wireless protocols can be used for P2P and P2I connections, such as WiFi-Direct, Long-Term Evolution (LTE), 5G, and 6G.

The transmission rate depends on factors like wireless channel bandwidth, transmission power, distance between devices, and channel interference. Using Shannon's theory [58], the transmission rate between devices u_i and u_j is given by:

$$I_{u_i, u_j}(t) = B \log_2(1 + \delta_{u_i, u_j}(t))$$

where B is the bandwidth, and $\delta_{u_i, u_j}(t)$ is the signal-to-noise ratio (SNR), computed as:

$$\delta_{u_i, u_j}(t) = \frac{(K \cdot p_{u_i, u_j} \cdot |h_{u_i, u_j}(t)|^2 \cdot d_{u_i, u_j}(t)^{-\alpha})}{N_0}$$

where K is a system constant, N_0 is the noise power, α is the path loss exponent, p_{u_i, u_j} is the transmission power, $d_{u_i, u_j}(t)$ is the distance between devices, and $h_{u_i, u_j}(t)$ represents the channel fading [59]. SNR can be also extended to signal-to-interference-plus-noise ratio (SINR) [60]. The total transmission latency is calculated as:

$$l_{u_i, u_j}(t) = \frac{V_i}{I_{u_i, u_j}(t)} + \frac{O_i}{I_{u_j, u_i}(t)}$$

where V_i is the task size, and O_i is the output size. Transmission time in wired channels is typically neglected due to their wide bandwidth and minimal latency [6].

4.2.3. Task model

A task is a computational unit executed remotely as part of an application component. Tasks may be independent or part of a workflow [24]. Several task models have been proposed. The two-tuple model represents a task with two components: $K_i = \{V_i, Y_i\}$, where V_i is the task size and Y_i is the Central Processing Unit (CPU) cycles per unit of data [41,49,51]. The tripartite model, often used in VEC and MEC, represents tasks as $K_i = (V_i, Y_i, D_i^{max})$, adding D_i^{max} for maximum tolerable delay [10,59,61–63]. Some studies use a variant where tasks are modeled as $K_i = (V_i, Y_i, O_i)$, with O_i representing output size in Kbs. The quadruple model combines these components into $K_i = (V_i, Y_i, O_i, D_i^{max})$ [64–66].

More complex task models have been proposed. In [67], tasks are represented by $K_i = (V_i, O_i, Y_i, B_i^{ul}, B_i^{dl}, D_i^{max})$, where B_i^{ul} and B_i^{dl} represent the required cellular traffic. In Directed Acyclic Graph (DAG) based models, tasks are represented by DAG $G = (S, F)$, where S is the set of subtasks and F is the set of data dependencies between them [8,68,69]. Deep Neural Network (DNN) offloading tasks are modeled as $\Gamma_n = (v_n, \rho_n, D_n^{max})$, where v_n is the input data size, ρ_n is the inference workload, and D_n^{max} is the latency requirement [9].

Tasks can have varying priorities depending on the application context, system requirements, and offloading objectives. Priorities can guide resource allocation by allocating more resources on high priority tasks [67], influence task scheduling [45], affect ordering and queueing [40], or determine an offloading target device [40,70]. At the application level, priority may be defined by the urgency of the task. For example, in medical applications, the importance of subtasks can vary depending on a patient's medical history [71], while in autonomous driving, safety-critical tasks are typically prioritized over entertainment-related functions [72]. Commonly used methods for assigning priority is based on task deadlines [40,45,67,72], computational requirements [73], and cost [74]. Another widely used approach is structural priority in workflows, where precursor tasks must be completed before their dependent tasks [71,75]. These priority schemes are typically static and predefined, making them straightforward to implement but may be insufficient for addressing the challenges of dynamic environments.

In more advanced approaches, task priority is expressed as a utility function, which is optimized instead of directly optimizing specific metrics such as task delay [72]. Furthermore, a multi-layer DRL-based framework has been employed to learn task priorities dynamically and schedule them based on their learned levels of importance, distinguishing between high- and low-priority tasks[40].

4.2.4. Computation model

The computation model defines available computation modes, such as local, P2P, edge, Edge-to-Edge (E2E), and cloud. In other words, it defines where the computation is performed. The most common modes are vertical offloading (edge, cloud) and P2P from horizontal offloading. Some studies extend horizontal offloading into E2E [41,42]. Computation models typically describe computational time, energy consumption, and constraints. In this section, simplified computation models are presented.

Local computation time is given by $ct_i^{local} = \frac{Y_i}{c_i}$, where c_i is the device's computing capacity and Y_i the CPU cycles per unit of data. Energy consumption is defined as $E_i^{local} = \kappa_0 H_i Y_i$, where H_i is the energy consumption per CPU cycle and κ_0 is the energy coefficient, which depends on the chip structure [49]. When offloading tasks to another device, transmission time must be factored into the total latency. Then, the total latency between u_i and u_j is given by $L_{u_i, u_j} = l_{u_i, u_j}(t) + ct_j^{local}$ when offloading a task from u_i to u_j . Total energy consumption is then calculated as:

$$E_{u_i, u_j} = E_{u_i, u_j}^t(t) + E_{u_i, u_j}^r(t) + E_j^{local} + E_{u_j, u_i}^t(t) + E_{u_j, u_i}^r(t)$$

where $E^t(t) = \frac{V_i}{I_{u_i, u_j}(t)} P_{u_i, u_j}^t$ is the energy to offload a task, $E^r(t) = \frac{V_i}{I_{u_i, u_j}(t)} P_{u_i, u_j}^r$ is the energy to receive a task, and E_j^{local} is the energy to perform the task. P is the transmission power constant.

For multi-hop offloading, the total latency is the sum of transmission latencies from device u_i to u_j through intermediate nodes u_k :

$$L_{u_i, u_j} = \sum_{k=i}^{j-1} l_{u_k, u_{k+1}}(t) + ct_j^{local}$$

and the total energy consumption:

$$E_{u_i, u_j} = \sum_{k=i}^{j-1} (E_{u_k, u_{k+1}}^t(t) + E_{u_k, u_{k+1}}^r(t)) + E_j^{local} + \sum_{k=j}^{i-1} (E_{u_k, u_{k+1}}^t(t) + E_{u_k, u_{k+1}}^r(t))$$

The computing model also establishes constraints, including maximum delay tolerance, transmission power, energy consumption, and computational capacity.

4.2.5. Mobility model

Previous surveys have reviewed various mobility models used in task offloading and service placement in MEC systems. In a survey on mobility-aware offloading [19], the authors discussed mobility models, including stochastic models and random walk. Mobility can be modeled with time slot structures, where users' positions are fixed within each time slot but change randomly between slots [45]. Vehicles are often modeled based on their relative positions and velocities with respect to the RSU, where the maximum communication time is derived from their remaining distance [63]. Some works assume mobility follows a Gauss-Markov process to generate trajectories [76], while others use synchronized random walk models to simulate vehicle distribution and speeds [77]. Real-world mobility datasets are a common way to simulate mobility of device. Datasets have been collected from many cities, such as Cretey, France [56], GAIA dataset from Xi'an China [78], and Chengdu, China [59]. More mobility datasets from different domains can be found from [79]. The ONE simulator [80] is also used to capture real-world mobility patterns in network simulations [81]. For vehicle mobility, kinematic models update positions using velocity and acceleration data [60]. Mobility models can be deployed to synthetic environments or real-world mobility simulations, such as those developed using SUMO [57].

4.3. Overview of computation offloading

Offloading refers to the transfer of data, including application code, processes, or requests, from one device to another. This section covers the fundamentals of computation offloading, focusing on how offloading problems are formulated, the objectives it aims to achieve, and the proposed approaches for solving the problem.

4.3.1. Problem formulation

Task offloading is typically framed as an optimization problem. It can be expressed as a constrained single-objective [81] or multi-objective optimization problem [46]. Often, multi-objective problems are transformed into a single-objective by combining objectives into a single weighted function [41,45]. Various joint optimization problems have been explored, such as task offloading [46], resource allocation [46], power allocation [82], resource scheduling [83], and communication decisions [47]. Hard constraints are typically application-specific and include limitations such as latency bounds [41,65], bandwidth and computational capacity limits [41,57,65], energy constraints [69], and the requirement to select exactly one offloading mode [57,84].

Task offloading optimization problem is often NP-hard [10,45,52,60,85,86]. One common formulation is Mixed-Integer Nonlinear Programming (MINLP) [10,60,84,85], which involves both integer and continuous decision variables with nonlinear objective functions and/or constraints. Several optimization problem types are found in the literature, including stochastic [38,87], long-term [40], multi-slot stochastic [57], nonlinear integer programming (NIP) [39,82], and integer programming [69], all of which are NP-hard.

4.3.2. Optimization objectives

Numerous objectives have been proposed, including minimizing energy consumption [62,88], latency [46,68,69], task completion time [43,52], reliability [89], throughput [68], computing resources [46,86,90], transmission rate [85], computation rate [42,59,91], social trust [7], satisfaction [8], and cost [38,65,66,78,92]. Cost may include various metrics, such as communication cost [92], energy consumption cost [48,49], and processor utilization cost [92]. Common objectives in joint optimization with task offloading include energy consumption [69,87]

and resource allocation [64,90,93]. In UAV-MEC networks, the UAV trajectory is often optimized jointly with other objectives [57]. These objectives can sometimes be conflicting, requiring a trade-off. For example, in the relationship between energy consumption and latency, increasing transmission power can reduce latency but simultaneously lead to higher energy consumption [60].

4.3.3. Task offloading approaches

A range of methods have been proposed to address the offloading optimization problem. In this article, we classify these methods into four categories: classical, heuristic, metaheuristic, and machine learning-based approaches. Often, these approaches integrate methods from different categories into a unified framework. In this work, we classify the approaches based on the technique employed to solve the task offloading problem.

Classical methods are well-established, deterministic approaches that guarantee optimal solutions for convex problems. However, as offloading is a complex, often NP-hard problem, it poses significant challenges in terms of computational feasibility. To overcome this, the original problem is typically decomposed or transformed into more manageable subproblems. These subproblems can be solved either iteratively [91] or sequentially [38,60]. Key techniques in classical methods include Dynamic Programming (DP) [89], the Alternating Direction Method of Multipliers (ADMM) [49,60,94], Lyapunov optimization [38,82], and the Karush-Kuhn-Tucker method (KKT) [7]. These methods are commonly applied in less complex collaborative systems [5].

Heuristic methods are problem-specific strategies based on practical experience or domain knowledge. While they do not guarantee optimal solutions, they provide practical approximations quickly. Similar to classical approaches, the original problem is often transformed into subproblems which are solved by heuristic methods [7,67,84]. Common heuristic methods include greedy algorithms [59], matching-based approaches [67,81,90], and search methods [9,62].

Metaheuristic methods, which have gained increasing popularity, can be applied to a broad range of optimization problems. Unlike heuristic methods, they offer a generalizable framework that is not tailored to a specific problem. They are particularly suited for large-scale and complex issues, though they may require longer solution times. Common metaheuristic methods include game theory [8,44], and nature-inspired methods, such as Particle Swarm Optimization (PSO) [39,52], Genetic Algorithms (GA) [61,95], and Ant Colony Optimization (ACO) [10,63,64].

Machine learning methods have been applied to various problems with promising results, especially in complex and dynamic environments, making them popular especially in VEC networks. The optimization problem can be reformulated as a Markov Decision Process (MDP) and solved using RL and DRL methods. DRL methods are adaptable to various problems, especially those involving uncertainty or incomplete information, though they may require large datasets and long training times, and their solutions may lack interpretability. Common algorithms include Q-learning [55], Deep Q-learning (DQN) [56,86], Deep Deterministic Policy Gradient (DDPG) [47,53], Dueling and Double DQN (D3QN) [40], and Proximal policy optimization (PPO) [42]. MARL extends RL methods to multiple agents, such as Multi Agent DDPG (MADDPG) [57]. For more detailed coverage of RL methods in offloading, see [28–30].

Offloading approaches can be further categorized into centralized, decentralized, and distributed methods. The optimization problem can either be formulated as a global problem, where the dynamics of the entire system are optimized, or as a local problem, where the dynamics of individual agents are optimized. Methods that address global optimization problems within a central unit and have access to complete system information, are classified as centralized approaches [10,52]. Some methods traditionally regarded as distributed are classified as centralized when implemented in a centralized manner. In contrast, if a global optimization problem is solved by aggregating results of local actions, it

is classified as distributed. For instance, PSO and ACO are examples of distributed methods used to solve global optimization problems [39,63]. Decentralized approaches, on the other hand, are deployed on individual agents to solve local optimization problems without any central coordination. In this article, decentralized methods refer to those where the optimization problem is solved on each end device, rather than at the edge layer, as tasks are generated by the end devices. RL often combines both paradigms, with centralized training followed by decentralized inference [96].

5. Task offloading approaches in collaborative computing: Categorization and comparison

This section describes the collaborative systems and task offloading approaches proposed in the selected studies. The works are categorized by offloading approach and further analyzed according to system architecture, collaborative characteristics, offloading dynamics, and evaluation methodology, with a summary of the findings provided afterward. Due to the large number of comparison methods and unique acronyms/abbreviations (103 in total) presented in Tables 2–5, the full names have been omitted for clarity. Readers are kindly referred to the original sources for detailed explanations.

5.1. Classical approaches

Various classical methods have been proposed to address the task offloading problem, including Lyapunov optimization [38,82,87], KKT method [88], Lagrange duality [91], Lagrange multipliers [60], Gradient sampling [97], DP [89], and the Gauss-Seidel method [82]. Some studies have proposed hybrid algorithms combining multiple techniques. For instance, the UATO algorithm, based on Lyapunov optimization, uses the Gauss-Seidel method to solve user association and power allocation [82]. Similarly, the Steerable Economic ExPense (SEEP) algorithm, proposed in [38], utilizes Lyapunov optimization and iterative algorithms to solve subproblems with closed-form solutions.

Many studies focus on joint optimization, where task offloading is optimized alongside other factors such as resource allocation [38,91], power control [60], and energy transmission [91]. These joint optimization problems are often decomposed into separate subproblems, which are solved iteratively [91] or sequentially [38,60] using various techniques. Common optimization objectives include minimizing cost [97], energy consumption [49,60,87,88], and latency [60].

Several studies adopt a centralized approach to solve the offloading problem [82,87,88,91]. In contrast, the ADMM distributed optimization method, which decomposes the global problem into smaller subproblems that each node can independently solve, has been used in [49,60]. Furthermore, some works formulate the problem as a local optimization task, or task assignment is decided on the agent-level in a decentralized manner [38]. A summary of the approaches, objectives, and methods for comparative evaluation is provided in Table 2.

These approaches are typically applied in relatively simple MEC and VEC systems, where task offloading occurs primarily between end devices. Among the reviewed works, only [99] considers a UAV-MEC scenario. Some works extend the approach to vertical offloading, incorporating edge servers [49,60,87]. In VEC systems, these methods are also extended to handle more complex scenarios [60,89]. However, classical approaches tend to overlook complex intra-layer dynamics, such as task dependencies, multi-hop offloading, and mobility. While some studies consider mobility [60,82,89], and multiple network hops [89,97], only one study considered dependent tasks [98], as reflected in Table 6. Evaluation of these methods typically involves numerical simulations, using tools like MATLAB [87] and Python [60], with environments ranging from small-scale simulations involving 14–60 end devices [60,82,87,88,91,97] to larger-scale simulations with up to 200 end devices [49], as illustrated in Fig. 4. In [98], the authors developed a hybrid evaluation platform that combined real devices, such as Raspberry

Pis and Jetson Nanos, with the CARLA simulation environment and datasets covering scenarios ranging from surveillance to traffic. Many approaches were compared to methods with limited capabilities, such as those relying solely on local computing [49,91] or optimizing only a subset of the problem [38].

5.2. Heuristic approaches

Heuristic methods for task offloading involve strategies that use problem-specific rules or domain knowledge to quickly find practical solutions, although these solutions may not be optimal. Common heuristic methods include greedy algorithms [59], Monte Carlo tree search [69], graph-based approaches [71], and matching-based methods [7,50,67,81,90]. Some studies propose hybrid approaches that combine multiple methods. For instance, the authors in [84] introduce JORA scheme, where the problem is decomposed into three subproblems and solved using an iteration-based algorithm and a game theory-based distributed algorithm. Similarly, Cui et al. [65] proposes the ICCRA system, which uses a heuristic algorithm for vehicle selection, while resource allocation is addressed through reinforcement learning.

Heuristic methods commonly address joint optimization problems. Common joint optimization challenges included resource allocation [9,46,65,84,90,108], task partitioning [68], and task assignment [69]. The typical optimization objectives in this category are energy consumption [7,9,62,67,69,81,84,108], latency [46,65,68,69,84], and cost [50,71,84,92]. Some works also consider trade-offs [84] and propose novel objectives, such as social trust [7]. A summary of the approaches is provided in the Table 3.

Approximately half of the methods in this category utilize a centralized approach to solve the optimization problem. In decentralized methods, the objective function is formulated at the agent level [46,71,108], with each agent making offloading decisions based on local information [59,65]. Distributed methods, on the other hand, apply game-theoretical [84] and pricing-based [7] approaches, where each agent participates in solving the global optimization problem.

Heuristic methods are widely applied in more complex MEC and VEC systems, and one study focusing on UAV-MEC [46]. Most approaches in this category involve vertical offloading between end devices and edge servers, with a few works extending collaboration to the cloud [65,67,84]. Notably, in [65] full computing continuum is utilized, enabling horizontal offloading in both the end and edge layers. These methods also address more complex intra-layer dynamics, such as mobility, multi-hop networks, and task dependency [68,69,90,92], as shown in the Table 6.

Evaluation is primarily conducted through numerical simulations, with environments typically involving 4–100 end devices. Some works use larger-scale simulations with 300 [50], 500 [90], and 1000 end devices [81] in MEC systems and 140 end devices [59] in VEC systems. Some studies incorporate simulators for more realistic evaluation like SimuLTE [126] in [67] and ONE [80] in [81]. Numerical simulations were supplemented with a real-life mobility dataset [59], where the authors utilized 180 vehicular trajectories from 2014 in downtown Chengdu, China.

5.3. Metaheuristic approaches

Metaheuristic methods have gained significant popularity in recent years due to their ability to address more complex optimization problems that arise in collaborative task offloading. Nature-inspired approaches, such as swarm intelligence and evolutionary algorithms, are dominant in this category. Swarm intelligence methods like ACO [10,63,64], PSO [39,52,115], and Artificial Bee Colony (ABC) [45] are commonly used, along with evolutionary algorithms such as GA [95,121], Cuckoo-search [125], and Biogeography-based Optimization (BBO) [51]. Also, game-theoretical methods have been proposed [8,44]. Swarm intelligence methods are inherently decentralized or distributed

Table 2
Classical approaches.

		Computing mode	Algorithm	Optimization	Objective method	Comparison	
[82]	2017	MEC	Local, D2D	UATO Algorithm based on Lyapunov optimization. Gauss-seidel method for user association and power allocation.	Task offloading, transmit power allocation, D2D user association	Time-averaged network power consumption minimization	–
[88]	2018	MEC	Local, D2D	An energy-efficient task scheduling (MEETS) algorithm. KKT to solve offloading problem.	Task offloading	Maximize the energy efficiency	ETO; EETO
[38]	2018	MEC	Local, D2D	SEEP Algorithm based on Lyapunov optimization. Power allocation: iterative algorithm. Task scheduling, subcarrier assignment: closed forms.	Task offloading, resource allocation	Minimize the average expenses of mobile devices	OT; JOSP
[91]	2019	MEC	Local, D2D	Alternating-optimization-based approach. User-helper pairing via exhaustive search and Greedy-Based pairing.	Task offloading, resource allocation, energy transmit	Maximize the sum computation rate	Local; Uniform beamforming
[49]	2023	MEC	Local, D2D, Edge	ADMM-based Offloading and Resource Allocation Algorithm	Task offloading, Resource allocation	Minimize the overhead of the energy consumption and resource cost	Random; Distance-based
[87]	2024	MEC	Local, D2D, Edge	A Lyapunov based resource allocation and offloading (LMDRACO) algorithm; Submodular methods and convex decomposition to solve subproblems.	Task offloading, green energy scheduling decisions	Minimize energy consumption, execution latency, and energy waste	LRE; ESE; RDE
[97]	2024	MEC	Local, D2D	A nested-loop algorithm & a single-loop algorithm using the tools of gradient sampling and online mirror descent.	Workload allocation, Routing	Minimize cost, maximize the total network utility	SGP
[89]	2021	VEC	Local, V2V	Reliability-Oriented Offloading using Dynamic programming method	Task offloading	Maximize reliability	Random; Dynamic; Uniform1-2; DP
[60]	2022	VEC	Local, V2V, Edge	Offload decision by ADMM, Resource allocation by dual decomposition with duality and subgradient methods, Power control with Lagrangian multiplier method.	Task offloading, resource allocation, power control	Trade-off between energy consumption and latency	Greedy Approach; Exhaustive Search; Centralized
[98]	2025	VEC	Local, V2V, Edge	Offloading and partition algorithm based on the HLPP	Task offloading, task partitioning	Minimize the completion latency and energy consumption	Local; Edge; Neuro; DADS
[99]	2025	UAV-MEC	Local, D2D, Edge	An iterative algorithm based on penalty successive convex approximation (P-SCA) and Lagrangian duality	Task offloading, caching, UAV trajectory planning, resource allocation	Minimize system average latency for task completion	UAV-only; CFS; NDO; CTS; RCA

[39,44,45,48,51,63], with some methods implemented in central manner [10,52]. In centralized methods, the edge node acts as a global controller to allocate tasks across the network [52,64,115]. Other centralized methods include GA [95,121] and Simulated Annealing-based ANSSA [93]. Joint optimization problems in this category involve resource allocation [10,51,61,63,64,93], task partitioning [39], and matching decisions [51]. Common optimization objectives include energy consumption [10,44,45,51,63,64,93,115], latency [39,44,45,51,95], and cost [61,93]. In [8], the authors proposed a novel satisfaction-based objective. A summary of the approaches is provided in the Table 4.

Metaheuristic methods are typically used in more complex collaborative systems, including both end and edge-layer horizontal offloading along with vertical offloading. These methods also address intra-layer dynamics, such as multi-hop offloading and task dependency. Task dependency in workflow applications are often modeled using DAGs [8,45]. These methods were primarily evaluated using numerical simulations, which were implemented using MATLAB [45,51,64,93] or Python [8,10]. Many studies also incorporated mobility datasets [44,61]

and device capability data [64,93], including parameters such as channel gain, computing power, and task data size. Utilized mobility datasets include the HighD dataset [61] and a variety of datasets from the cities of Melbourne, Shanghai, and Darmstadt as presented in [44]. TGFF was employed to generate workflow applications to simulate task dependency [44,115]. Large-scale simulation environments were used for evaluation, with up to 1000 end devices in MEC systems [48,115] and 100 end devices in VEC systems [52].

5.4. Machine learning approaches

Machine learning, especially RL and DRL, has become one of the leading methods for addressing task offloading problems. The offloading problem is reformulated as MDP and solved using a RL approach. A wide variety of RL methods have been proposed, such as Q-learning [55], DQN [56,66,78,86,142], Double DQN (DDQN) [144], D3QN [40], DDPG [47,53], PPO [42,85], and Asynchronous Advantage Actor-Critic (A3C) based algorithm [130]. An example DRL schematic is shown in

Table 3
Heuristic-based methods.

		Computing mode	Algorithm	Optimization	Objective method	Comparison	
[81]	2017	MEC	Local, D2D	A graph-matching-based optimal task assignment policy	Task assignment	Minimize the overall energy consumption of the task executions	Greedy; Reciprocal; Random
[90]	2018	MEC	Local, D2D, Edge, cloud	A socially-aware cooperative offloading algorithm: a minimum-weight bipartite perfect matching	Task offloading, Resource allocation	Minimize the system overhead	Socially-Oblivious; Cloud; Local
[84]	2019	MEC	Local, D2D, Edge, Cloud	JORA scheme. RA solved by Iteration-based algorithm. Offloading mode selection by distributed joint optimization with game theory adaptation.	Task offloading, Resource allocation	Minimize the system cost; a trade-off between task processing delay and energy consumption	DCOA [100]; RND; MEC W. Local [101]; MEC W. D2D [94]
[50]	2019	MEC	Local, D2D	A Stackelberg game based incentive mechanism and optimal task assignment with a graph-based algorithm	Task offloading	Maximize utility function based on resource price	Greedy match; Random fixed price
[69]	2020	MEC	Local, D2D, Edge	SAHCO framework. TA-MCTS for task assignment	Task offloading, task assignment	Minimize energy consumption and delay	EOS; RCAS; TA-MCTS/SA; MWBMS
[67]	2020	MEC	Local, D2D, Edge, Cloud	Energy efficient and delay-aware offloading scheme (EEDOS). Offloading decision by maximum matching with minimum cost graph algorithm	Task offloading	Minimize the energy consumption	MEC selective [102]; D2D crowd [103]
[7]	2022	MEC	Local, D2D, Edge	The Lyapunov-based problem transformation. Short-term problems solved by the KKT method and a pricing-based matching algorithm	Task offloading	Minimize energy consumption and maximize social trust	Four heuristic methods
[9]	2023	MEC	Local, D2D, Edge	Two-level optimization algorithm. TO: Heuristic, RA: a subgradient based algorithm.	Task offloading, Resource allocation	Minimize energy consumption	GA [104]; ABAA; EOA; NDA
[71]	2024	MEC	Local, D2D	DCO-HM framework. DAG-based offloading algorithm.	Task offloading	Optimize the collaborative system cost	PADGTO [105]; DTOA [106]
[92]	2020	VEC	Local, V2V, Edge	Mobility-Aware partial (MAP) task offloading algorithm	Partial task offloading	Minimize total cost	Conventional partial; MEC partial offloading [107]
[65]	2021	VEC	Local, V2V, Edge, Cloud	ICCRA system. KNN for layer selection, Heuristic for collaborative vehicle selection. Resource allocation solved by multi-objective RL.	Task offloading, resource allocation	Minimize latency and transmission rates	Local; Edge; Random
[62]	2021	VEC	Local, V2V, Edge	A multi task offload strategy. Search-based method.	Task offloading	Reduce energy consumption	Local; no priority
[59]	2021	VEC	Local, V2V, Edge	A real-time distributed strategy (RtDS). Decision-making by dynamic programming-based method and a minimum-delay-greedy method.	Multi-period task offloading	Maximize the sum computation rate	Cent. & Dist. Heuristic; RtDS-maxD
[68]	2022	VEC	Local, V2V, Edge	Application-centric offloading framework based on application partitioning.	Task offloading, Task partitioning	Throughput, execution time	No partitioning methods
[108]	2024	VEC	Local, V2V	NOMA with a multitask offloading model using Optimal Stopping Theory (OST)	Task offloading and resource allocation	Energy efficiency	OMA; NOMA; NOMA with SIC
[46]	2024	UAV-MEC	Local, D2D, UAV	Alternately iterating subproblems (RA, TO) solved by KKT	Task offloading, resource allocation	Minimize the average computing resources, and maximum delay	Average

Fig. 3. Additionally, MARL approaches have been explored to address task offloading problems [57,85], such as MADDPG in [57]. Both DRL and MARL methods have been further improved through the use of attention mechanisms, which help address uncertainty [70,135], and accelerate convergence [135]. Beyond these, recent studies have investigated the potential of Graph Neural Networks (GNN) [146] and

Graph Reinforcement Learning (GRL) [134], where network topologies are modeled as graphs, with edges representing connections and nodes encoding the state information of computational units.

The design of the RL method determines whether it is centralized or decentralized. In many cases, training involves some degree of centralization, where data is exchanged between agents to refine the model.

Table 4

Metaheuristic methods.

		Computing mode	Algorithm	Optimization	Objective method	Comparison	
[64]	2020	MEC	Local, D2D, Edge	A bilevel optimization method that integrates Ant Colony Algorithm	Task offloading, Resource allocation	Minimize the total energy consumption	Local; MEC-C; CC
[48]	2021	MEC	Local, D2D, Edge	A stackelberg game embedded into Lyapunov optimization for task offloading. Mean field model for D2D collaboration.	Task offloading, D2D collaboration	Optimize the longterm utility: average profit and processing cost.	Constant PO; Adapted PO
[93]	2022	MEC	Local, D2D, Edge	An adaptive neighborhood search based on simulated annealing algorithm (ANSSA) for joint optimization	Task offloading, resource allocation	Minimize the sum of calculation costs and energy consumption	BIJOR [109]; IANSSA
[8]	2022	MEC	Local, D2D	Game theory based workflow offloading strategy with no-regret dynamics	Task offloading	Maximize satisfactory, defined as deadline, budget, reward, and reliability	DCL [110]; BR [111]; TS [112]; PSO [113];
[44]	2022	MEC	Local, D2D, Edge, Cloud	Distributed Game-Theoretical Offloading Algorithm	Task offloading	Each user minimize latency and energy	BRUO; BUAU; COTO; RTO [114]
[95]	2023	MEC	Local, D2D, Edge	An improved GA for distributed multi-hop offloading	Task offloading	Minimize task completion delay	Local; D2D-Fogging; Min-Min Algorithm
[115]	2024	MEC	Local, D2D, Edge	Incremental control and swarm intelligence-based techniques for task offloading	Task offloading	Minimizing the whole energy dissipation of end devices	ELYO [116]; ELDM [117]; ELGT [118]; ERPS [119]; EILP
[45]	2024	MEC	Local, D2D, Edge	HEFT algorithm for dependent task scheduling. A Hybrid artificial bee algorithm for offloading.	Task offloading	Minimize all request users' delay and energy consumption	DE; GA; ABC; AWPSO; RWOA
[39]	2024	MEC	Local, D2D, Edge	PSO algorithm for scheduling decisions, DP algorithm to find the optimal partition	Task partitioning, task scheduling	Minimize the sum of all task devices' maximum latency	noD2D; G-PSO; R-PSO
[10]	2024	MEC	Local, D2D, Edge	Two-level alternating optimization framework: ACO for population of offloading decisions, and DDPG for optimal decision.	Task offloading, resource allocation	Minimize the energy consumption of the entire end-devices	ACO-AC; OMA; BOA; WC; LE
[120]	2025	MEC	Local, D2D, Edge	HAJTORA: the improved AL-SHADE algorithm with mathematical methods	Task offloading, Resource allocation	Minimize the total system cost	Local; D2D Disabled; Random
[121]	2025	MEC	Local, D2D, Edge	IGA-DDPG: a two-level optimization model based on Improved GA and DDPG	Task offloading, Resource allocation	Minimize the total system cost	IGWO; PNCM; binary; Local
[51]	2023	VEC	Local, D2D	One-to-many matching by an improved BBO (IBBO), low-complexity method for one-to-one matching.	Matching decisions, resource allocation	Minimize the latency, and the energy consumption	FLCS; FOS
[61]	2023	VEC	Local, V2V	BENEFIT. Domain Generalization for trajectory prediction. Enhanced Non-Uniform genetic algorithm for offloading decisions.	Task offloading, Resource allocation	Minimize task completion cost of all vehicles	Local; Greedy; GA
[52]	2023	VEC	Local, V2V, Edge, Cloud	IoVTA-IQPSO, Quantum PSO algorithm	Task allocation	Maximize task completion utility	Greedy algorithm; common PSO
[63]	2024	VEC	Local, V2V, Edge	EJSOC scheme. Consists Ant colony selection for selection decision, and Lagrange multipliers and gradient methods for resource allocation.	Task offloading, Resource allocation	Minimize energy consumption	COO [122]; RCO [123]; EON; FCR [124]
[125]	2025	VEC	Local, V2V, Edge	An improved discrete Cuckoo search algorithm	Task scheduling, Cache updating	Maximize the offloading efficiency	DTOSC; TO-V2I; To-w.o -R; To-w.o -C

However, after training, the model can be deployed in a decentralized manner, with each agent making decisions based on its local information. In certain cases, the optimization function, and MDP, is designed to optimize a global objective, which necessitates a centralized view of the system [41,43,47,53,78,142,144]. Additionally, in [77] the authors

proposed distributed adaptive task offloading framework using RL. A summary of the algorithms is provided in Table 5.

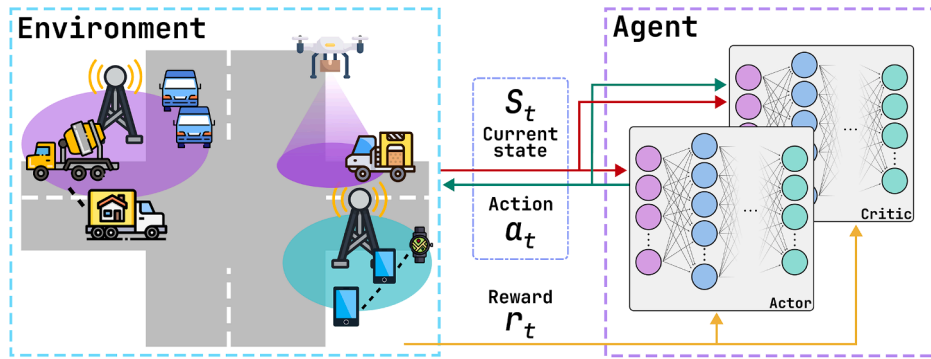
Machine learning methods have been applied primarily in VEC systems, although some studies also explore MEC. In [57], task offloading problem in UAV-assisted VEC (UAV-VEC), where vehicles, edge servers,

Table 5
Machine learning methods.

		Computing mode	Algorithm	Optimization	Objective method	Comparison	
[55]	2022	MEC	Local, D2D	Q-learning for offloading and resource management.	Task offloading, resource management	Trade-off between system operation cost and task execution time	Local; Greedy offloading
[54]	2023	MEC	Local, D2D, Edge	Attention Communication Deep Reinforcement Learning (ACDRL) for offloading decisions	Task offloading	Minimize long-term delay	MADDPG[127]; MAAC-based [128]; DQN [129]
[130]	2023	MEC	Local, D2D, Edge	A swap-matching-based algorithm for optimal subchannel allocation, A3C algorithm to optimize the rest of the resource variables.	Task offloading, subchannel allocation, power control, resource allocation	Minimize execution latency and energy consumption while guaranteeing the secrecy	DQN [131]; (DPPO) ² -based [132]
[42]	2024	MEC	Local, D2D, Edge	PPO algorithm for offloading and resource allocation	Task offloading, resource allocation	Minimize task execution time and energy consumption	Random; DQN [133]; A2C; PPO
[41]	2024	MEC	Local, D2D, Edge, cloud	DJTRA algorithm; an SD3-based method for task offloading. Various numerical methods for others.	Task offloading, transmission power allocation, resource allocation	Minimize the total system cost considering the time and energy consumption	DTOD; EECD; CPRAD; CMRAD
[70]	2024	MEC	Local, D2D, Edge	DMOEA: DRL with entropy and attention mechanism, multi-actor single-critic structure	Task offloading	Minimize the total latency of all tasks	Local; Edge; DQNCCO; SAC-CO
[134]	2024	MEC	Local, D2D, Edge	GCTO: A collaborative task offloading scheme based on GRL	Task offloading	Minimize average energy consumption	Local; Random; ETOS; SAC-CO
[75]	2025	MEC	Local, D2D, Edge, Cloud	RASO: A redundancy-aware adaptive search offloading based on DQN	Task offloading	Minimize the total time delay and energy consumption	TOERT; BR-DMCTO; SDTO; COFE; OSTTD
[135]	2025	MEC	Local, D2D, Edge	An attention-enhanced MADDPG (AT-MADDPG)	Task offloading	Maximize the offloading benefit	MEC only; D2D only
[78]	2020	VEC	Local, V2V, Edge	Data scheduling solved by DQN algorithm.	Data scheduling	Minimize the system-wide data processing cost	Local; RSU; No collab.
[85]	2021	VEC	Local, V2V	MARL framework with new reward function. V2V policy learned by PPO.	Task offloading, power allocation, spectrum reuse	Maximies QoE, expressed as delay and data rate.	Single-agent; Max-throughput; DQN
[86]	2021	VEC	Local, V2V, Edge	DQN	Task offloading, caching decisions, resource allocation	Minimize system overhead	Full Local; Full Offload
[56]	2021	VEC	Local, V2V, Edge	DQN based on Simulated Annealing (SA-DQN) algorithm	Task offloading	Minimize failure rate and the total energy consumption	Local; Edge; V2V; Sarsa; Q-learning
[53]	2022	VEC	Local, V2V, Edge	DDPG-OCWB algorithm for task offloading/caching and bandwidth allocation	Task offloading, task caching, bandwidth allocation	Minimize the system cost (i.e., system delay)	NoRP [136]; DGO; NoRC [137]; NoCV [138]
[77]	2022	VEC	Local, V2V	Adaptive task offloading framework with reward-based offloading scheme	Task assignment	Minimize offloading delay for task set	RWA [139]; UCB [140]; ALTO [141]
[142]	2022	VEC	Local, V2V, Edge	A hybrid task offloading scheme (HyTOS). DQN for optimal offloading decision-making.	Task offloading	Jointly optimize the delay and energy consumption	Local; single scenario; GT-hybrid [143]
[43]	2023	VEC	Local, V2V, Edge, Cloud	ORAD algorithm, based on the DDPG	Task offloading, resource Allocation	Maximize the success rate of task offloads	Local; V2V; V2I; Greedy strategy; Random; Probabilistic; PPO
[144]	2024	VEC	Local, V2V, Edge, Cloud	Graph Convolutional Networks with Double-DQN	Task offloading	Maximize resource utilization, while minimizing costs	DDQN; DQN; Q-learning; Local
[66]	2024	VEC	Local, V2V, Edge, Cloud	DQN-algorithm for optimal offloading decision-making	Task offloading, resource allocation	Minimize the system's total cost	Local; MEC; LMC; Random
[40]	2024	VEC	Local, V2V, Edge, cloud	A multi-layer DRL based approach; D3QN for offloading decision, PDQN for scheduling, classical optimization for resource allocation.	Task offloading, task scheduling, resource allocation	Optimize latency and energy consumption	LBA; RBA; GBA; SLDBA; MLDBA
[145]	2024	VEC	Local, V2V, Edge	Sensing-aware Joint service node Auction Offloading (SJAO) scheme. A DT-enhanced Twin Delayed Deep Deterministic Policy Gradient Perception Offloading (TD3PO) algorithm	Task offloading, resource allocation	Minimize the total system cost	TD3-Vanilla; The Auction-Classi; Random-BL

Table 5
Continued...

		Computing mode	Algorithm	Optimization	Objective method	Comparison	
[146]	2024	VEC	Local, V2V	Vehicular FedGNN: Federated Graph Neural Network (GNN) based offloading mechanism	Task offloading	Minimize the weighted latency	Non Edge; Non FL; Non FL&Edge; FedGCN
[47]	2025	VEC	Local, V2V, Edge, Cloud	MCCO-RAA: Two-tier Iterative Scheme Based on DDPG and Cooperative Game Theory to determine offloading decisions.	Task offloading, resource allocation, cache placement, communication decision	Minimize overall processing latency	COF-MMR [147]; LRU-GOP; LRU-AOESP; LRU-AOCSP; Hybrid-LCP
[148]	2025	VEC	Local, V2V, Edge	PORA-MATD3: MATD3-based strategy for optimal performance and TMS to handle short-term fluctuations.	Task offloading, resource allocation	Minimize the delay and energy consumption of task computation	DDPG; DQN; Local; All MEC
[149]	2025	VEC	Local, V2V, Edge	An asynchronous advantage actor-critic algorithm	Task offloading, resource management	Minimize the overall energy consumption	MOCIG; DQN-hybrid
[150]	2025	VEC	Local, V2V, Edge	A DRL approach based on the Rainbow algorithm	Task scheduling	Minimize the total service cost	DQN; DDQN
[57]	2024	VEC-UAV	Local, V2V, UAV, Edge	UA-MADDPG: MADDPG for offlaoding decision, uav flight trajectory, and spectrum resource allocation. GBCRA: resource allocation based on greedy algorithm.	Task offloading, resource allocation, UAV trajectory	Minimize the weighted system cost of energy consumption and delay	W/O UAV trajectory opt.; W/O Spectrum allocation; W/O Comp. Allocation; DQN; DDQN

**Fig. 3.** Deep reinforcement learning-based offloading scheme.

and UAVs interacted, was discussed. The MADDPG algorithm was used to optimize UAV flight trajectories, offloading decisions, and spectrum allocation. Many methods in this category employ the full end-edge-cloud architecture, with horizontal offloading on the end layer, while some studies have extended horizontal collaboration to the edge layer [41,42,53]. These methods also heavily incorporate mobility and multi-hop offloading. Task dependency was addressed in [56], where temporal dependencies were modeled using DAGs.

Evaluation of these methods is mainly performed through numerical simulations, with some studies incorporating mobility datasets [56,78]. Examples include vehicular trajectories from Crete, France [56], and the GAIA dataset from Xi'an, China [78]. In [57], simulation environment was created from a realistic road scene from Di'anmen West Street, Beijing, using SUMO. In [42], the authors proposed a real-life testbed for MEC where Raspberry Pis were used as end devices, and personal laptops served as MEC servers. Evaluation environments for RL-based methods typically involve fewer end devices than those used in meta-heuristic or heuristic methods. Typically, evaluation environments in this category included 1–10 edge servers, and up to 100–150 end devices. However, in a recent study [75], the authors evaluated their approach using 300 edge servers and approximately 1500 end devices (4–6 IoT devices per MEC), representing the largest evaluation system among the selected studies.

5.5. Summary

The methods discussed in this survey present a range of approaches for solving the collaborative task offloading problem, with a clear distinction between the different categories.

- **Classical** methods are well-suited for simpler environments, such as those involving only end and edge layers, where they offer optimal solutions that are easy to implement, but tend to overlook more complex intra-layer dynamics. Moreover, 55% of the methods employ centralized control. Notably, one recent study [98] conducted its evaluation using datasets, the CARLA simulation platform, and real-life equipment, making it the most comprehensive evaluation scheme among all the reviewed articles.
- **Heuristic** methods are applied for more complex MEC, VEC, and UAV-MEC system, with 50% of them employing a centralized control architecture. The majority of studies focused on end-edge architectures, while some incorporated end-edge-cloud architectures. Mobility was accounted for in nearly all VEC environments; however, only a limited number of studies incorporated multi-hop offloading or task dependency. Numerical simulation is the dominant evaluation environment, but one study used real-life mobility dataset and one network simulator to enhance the realism of the evaluation.

Table 6

Comparison of articles published between years 2021–2025 based on collaborative systems, dynamics, and evaluation methodology.

	Year	Network			Horizontal		Vertical			Mob. ^c	Hop	Task	Control	Eval	
		MEC	VEC	UAV	End	Edge	End ^a	Edge ^b	Cloud						
[49]	2023	✓			✓		✓	✓		–	S ⁱ	–	Di	S	Classical
[87]	2024	✓			✓		✓	✓		✗	S	I	C	S,D	
[97]	2024	✓			✓		✓			✗	M	–	Di	S	
[89]	2021		✓		✓		✓			✓	M	–	De	S	
[60]	2022		✓		✓		✓	✓		✓	S ⁱ	–	Di	S	
[98]	2025		✓		✓		✓	✓		✓	S	D	C	S,D,R	
[99]	2025			✓	✓		✓	✓		✗	M	–	C	S	
[7]	2022	✓			✓		✓	✓		–	S,M	I	Di	S	Heuristic
[9]	2023	✓			✓		✓	✓		–	S ⁱ	–	C	S	
[71]	2024	✓			✓		✓			–	S ⁱ	D	De	S	
[65]	2021		✓		✓	✓	✓	✓	✓	✓	S	–	De	S	
[62]	2021		✓		✓		✓	✓		✓ ⁱ	–	–	C	S	
[59]	2021		✓		✓		✓	✓		✓	–	–	De	S,D	
[68]	2022		✓		✓		✓	✓		✓	–	D	C	S	
[108]	2024		✓		✓		✓			✓	S	I	De	S	
[46]	2024			✓	✓		✓	✓		–	–	I	De	S	
[48]	2021	✓			✓		✓	✓		✓	S ⁱ	I	De	S	Metaheuristics
[93]	2022	✓			✓		✓	✓		✗	S ⁱ	–	C	S,D	
[8]	2022	✓			✓		✓			✓	M	D	C	S	
[44]	2022	✓			✓	✓	✓		✓	✗	S,M	I	Di	S,D	
[95]	2023	✓			✓	✓	✓	✓		✓	M	I	C	S	
[115]	2024	✓			✓		✓	✓		–	–	D	C	S	
[45]	2024	✓			✓		✓	✓		✓	–	D	De	S	
[39]	2024	✓			✓	✓	✓	✓		–	S,M	D	Di	S	
[10]	2024	✓			✓		✓	✓		–	S ⁱ	I	C	S	
[120]	2025	✓			✓		✓	✓		–	–	I	C	S	
[121]	2025	✓			✓		✓	✓		–	S	I	C	S	
[51]	2023		✓		✓		✓	✓		–	–	–	De	S	
[61]	2023		✓		✓		✓			✓	–	–	C	S,D	
[52]	2023		✓		✓	✓	✓	✓	✓	✓	S,M	I	C	S	
[63]	2024		✓		✓		✓	✓	✓	✓	S,M	–	Di	S	
[125]	2025		✓		✓		✓	✓		✓	–	D	C	S	
[55]	2022	✓			✓		✓			✗	S	I	De	S	Machine learning
[54]	2023	✓			✓		✓	✓		–	–	–	De	S	
[130]	2023	✓			✓		✓	✓		–	–	–	De	S	
[42]	2024	✓			✓	✓	✓	✓		✗ ⁱ	S,M	–	De	R	
[41]	2024	✓			✓	✓	✓	✓	✓	–	S,M	–	C	S	
[70]	2024	✓			✓		✓	✓		–	S	I	De	S	
[134]	2024	✓			✓		✓	✓		✗	S	I	C	S	
[75]	2025	✓			✓		✓	✓	✓	–	M	D	C	S,D	
[135]	2025	✓			✓		✓	✓		✓	S	–	De	S	
[85]	2021		✓		✓		✓			–	S ⁱ	–	De	S	
[86]	2021		✓		✓		✓	✓		✓	S ⁱ	–	De	S	
[56]	2021		✓		✓		✓	✓		✓	S	D	De	S,D	
[53]	2022		✓		✓	✓	✓	✓		–	S,M	–	C	S	
[77]	2022		✓		✓		✓			✓	M	–	Di	S	
[142]	2022		✓		✓		✓	✓		✗	–	–	C	S	
[43]	2023		✓		✓		✓	✓	✓	✓	S	–	C	S	
[144]	2024		✓		✓		✓	✓	✓	✓	S	I	C	S	
[66]	2024		✓		✓		✓	✓	✓	✓	S ⁱ	–	De	S	
[40]	2024		✓		✓		✓	✓	✓	✓ ⁱ	S ⁱ	–	De	S	
[145]	2024		✓		✓		✓	✓		✓	–	–	C	S	
[146]	2024		✓		✓		✓			✓	S	–	Di	D	
[150]	2025		✓		✓		✓	✓		✓	S	–	C	S	
[47]	2025		✓		✓		✓	✓	✓	✓	M	I	C	S	
[148]	2025		✓		✓		✓	✓		✓	M	I	De	S	
[149]	2025		✓		✓		✓	✓		✓	S	–	C	S	
[57]	2024		✓	✓	✓		✓	✓		✓	S	I	De	S	

^a Local computing model.^b Includes all stationary and mobile edge servers, which are differentiated from end devices.^c Mobility. Mobile end devices.^d M = Multi-hop, S = Single-hop.^e I = Independent, D = Dependent.^f C = Centralized, De = Decentralized, Di = Distributed.^g Evaluation method. S = Simulation, D = Dataset, R = Real-life.^h – Not mentioned in the article.ⁱ Not mentioned in the text, but derived from the equations or algorithms.

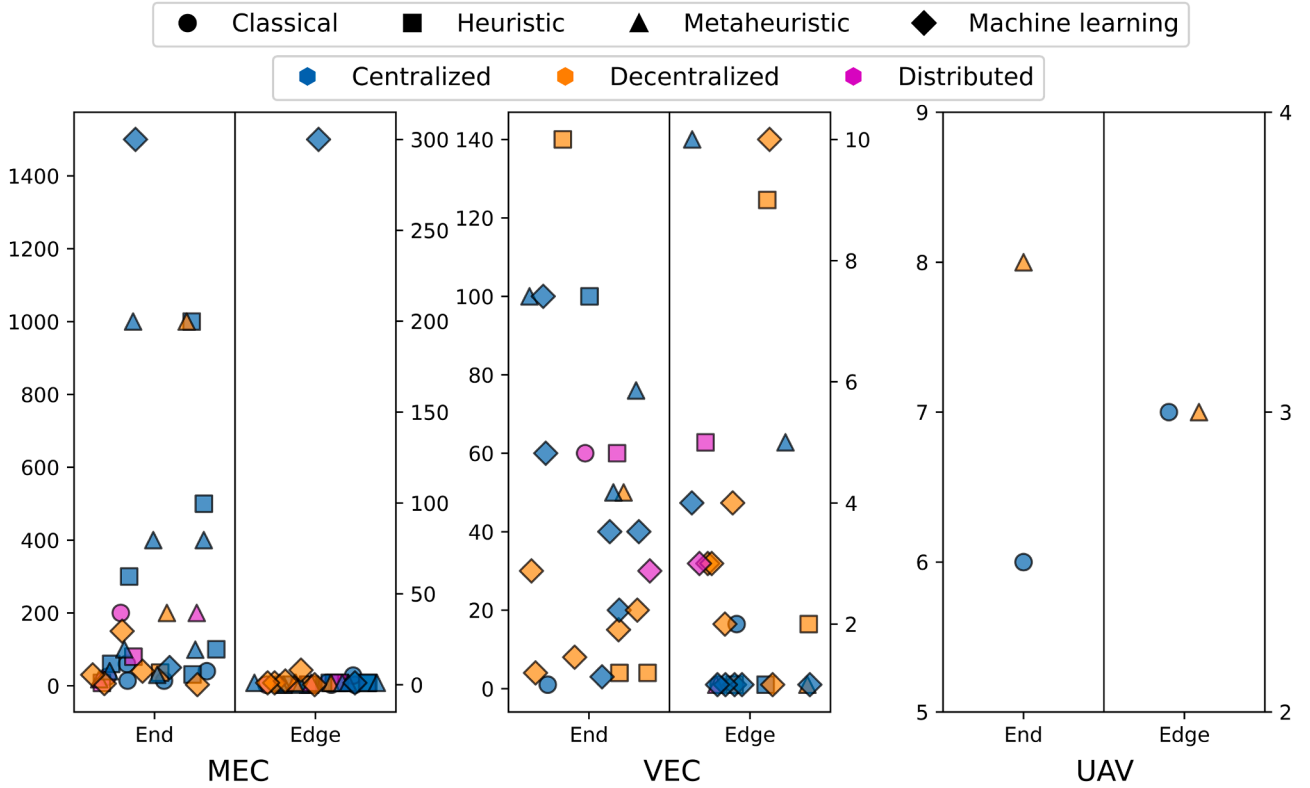


Fig. 4. Evaluation environment in the selected articles. Number of simulated devices at end-edge layers in MEC, VEC, and UAV-MEC/VEC.

- **Metaheuristic** methods, particularly nature-inspired approaches, are widely applied in complex systems involving dependent tasks and multi-hop offloading, with multiple layers of collaboration including full computing continuum. Although these methods are traditionally decentralized or distributed, 65% are implemented in a centralized manner. Numerical simulations and real-world mobility datasets were primarily used to create evaluation environments, scaling up to 1000 end devices and 10 edge servers. Additionally, a simulation tool was used to generate synthetic workflow applications for studying task dependencies.
- **Machine learning**, particularly RL methods, has garnered significant attention in recent years for its ability to generalize to unfamiliar environments and adapt to dynamic problems. Most studies applied machine learning methods to VEC networks, with one study focusing on UAV-VEC, where vehicles, edge servers, and UAVs interacted. Approaches were implemented in end-edge-cloud systems, with some studies incorporating horizontal collaboration at the edge layer. Additionally, many methods integrate multi-hop offloading with mobility, showcasing their ability to tackle complex problems in diverse environments. While 48% of the methods utilize decentralized decision-making during inference, centralization is often employed during training. Most evaluation environments rely on numerical simulations, with some integrating real-world mobility datasets. A recent study utilized a large-scale simulation involving 1200–1800 end devices and 300 edge servers, representing the largest setup among all reviewed works [75]. Notably, another study proposed a real-life testbed for evaluation [42].

Numerical simulations continue to be the predominant evaluation method, with many studies integrating mobility datasets and network simulators to enhance realism in experimental environments. However, the complexity of real-world environments, including the effects of obstacles on network conditions and device connectivity, is not fully cap-

tured. The evaluation environments were often study-specific, and while some studies used similar configurations, no standardized benchmarking system was employed. Furthermore, most approaches were compared only against methods within the same category or those with fewer capabilities, lacking proper cross-validation. Although some studies introduced random elements in the evaluation, such as random device distribution or location placement, it remains unclear how well these methods generalize to unfamiliar environments or different networks. For instance, none of the studies performed cross-comparisons between different networks, such as MEC and VEC. These limitations are reinforced by the lack of open-source code or datasets across all reviewed works, with only a few indicating a willingness to share data upon reasonable request [10,42,53,66,145,148]. The absence of open-source implementations and shared data makes it difficult to reproduce results, compare methods across studies, and advance research effectively.

Overall, among the 71 selected studies, 52% of the methods adopt a centralized control, 32% employ decentralized control, and 15% utilize distributed control. MEC systems were the most commonly studied (52%), followed by VEC (44%), while only 4% focused on UAV-MEC/VEC systems. In terms of offloading strategies, 11% employed horizontal offloading at the edge layer, 66% utilized vertical offloading between end and edge layers, and 14% implemented vertical offloading across the end–edge–cloud. A small portion of studies (6%) examined all horizontal and vertical offloading paradigms within end–edge–cloud systems, 20% addressed two of the three intra-layer dynamics, and only one study considered all three. Task dependency was more extensively covered by heuristic and metaheuristic approaches, whereas multi-hop offloading was predominantly explored in machine learning and metaheuristic methods. Only 17% and 27% considered task dependency and multi-hop offloading, respectively. While mobility is a dominant factor in VEC systems, it also plays a significant role in MEC systems due to user movements. However, only 22% of the studies about offloading in MEC systems considered mobility.

6. Open challenges and future research

This section highlights the challenges and potential areas for future research in computation offloading in collaborative computing systems. We emphasize the need for unified and standardized research across all edge computing networks, alongside the advancement of algorithms and techniques for complex collaborative systems that are adaptable to unknown and dynamic environments, and diverse networks.

6.1. Unification of the networks

Task offloading in MEC, VEC, UAV, and satellite-assisted systems shares several common characteristics, including mobile end devices, a mix of stationary and mobile edge servers, and hierarchical resource distribution. While end devices are mobile and equipped with sensors, the scale of movement speeds and system resources varies across networks. For instance, vehicles typically move faster than mobile phones, while UAVs move faster than ground robots. However, with the growing popularity of micromobility devices, the distinction between vehicles and mobile devices is becoming less clear, as these user-operated devices move at different speeds and ranges. Additionally, mobility is no longer limited to the ground due to the increasing adoption of consumer UAVs.

Most current studies focus on a single network, with limited work exploring hybrid edge computing environments. As a result, it remains unclear how effectively existing task offloading methods can be extended to hybrid networks encompassing a wide range of mobile devices, vehicles, and robots. The increasingly ambiguous distinction between heterogeneous mobile devices and vehicles further challenges conventional categorizations of edge computing paradigms, complicating the design, evaluation, and deployment of offloading strategies. For example, in systems involving micromobility devices (i.e. scooters or e-bikes) alongside wearable technology and smartphones carried by the drivers, it is unclear whether MEC or VEC based approaches are more appropriate. These devices may move at speeds comparable to larger vehicles but possess resource constraints similar to mobile phones. Although UAV-assisted approaches might offer relevant insights, existing studies have evaluated them only in small-scale environments, limiting their applicability to larger systems. These challenges highlight the need for research on hybrid edge computing networks, which current domain-specific approaches fail to adequately address.

To address these challenges, we propose unifying offloading research across various edge computing paradigms into a single field. This would shift the focus from specialized methods more towards generalized approaches suitable for hybrid networks that resemble more real-world heterogeneous computing networks. As a result, methods could be designed and evaluated in closer alignment with real-world requirements, thereby accelerating deployment. As an initial step, cross-comparisons of existing methods should be conducted using current simulation environments. These can be adapted to perform preliminary evaluation in hybrid scenarios, since essentially the commonalities across edge computing networks, such as speed, resources, and distance, serve as scaling factors. However, further research is needed, particularly work towards more complex and realistic collaborative systems (discussed in Section 6.2), and creating evaluation frameworks capable of realistically simulating hybrid networks and enabling proper cross-domain comparisons (discussed in Section 6.3).

6.2. Complex collaborative systems

Horizontal offloading in the end layer is frequently studied, but E2E offloading, particularly within three-tier architectures, has yet to be fully explored. E2E offloading offers distinct advantages in various scenarios, particularly when edge servers are subjected to high workloads and require load balancing, or when an end device moves beyond the coverage area of one edge server and enters the coverage area of another, where the output is subsequently returned to the end device. In the context

of horizontal offloading dynamics, some studies focus on task dependency, multi-hop offloading, and mobility. However, only one study, which uses a relatively simple D2D collaboration scheme, incorporates all of these dynamics. Some studies have explored the full computing continuum alongside multiple dynamics (i.e. multi-hop and mobility, or multi-hop and task dependency), but no study to date has combined all dynamics. Task dependency is primarily examined in systems utilizing D2D or end-edge collaboration, with one exception incorporating E2E collaboration.

Current research largely concentrates on simplified collaborative systems that fail to capture the complexity of real-world environments, thus possibly only applicable to small-scale or specialized scenarios. In practical deployments, computational tasks often constitute complex workflows that integrate both sequential and parallel operations. For example, neural network inference and computer vision pipelines are increasingly deployed on mobile devices to enable edge intelligence. Such complex systems require effective management of task dependencies to ensure efficient and reliable distributed execution, which have been studied in the context of scientific computing, where workflows are decomposed into tasks and scheduled on cloud [74] or edge-cloud [73] servers, however it remains largely unexplored in dynamic D2D offloading scenarios. In such environments, workflow management must also maintain reliable connectivity despite mobility, interference, and fluctuating network conditions. Robustness can be enhanced through data relaying among intermediate nodes, ensuring communication when direct links fail, and further strengthened by leveraging the full computing continuum, where data can be transmitted through the cloud in extreme cases. As these dynamics are inherently interdependent, they should be addressed collectively to enable adaptive, resilient, and practically deployable task offloading solutions.

To address these challenges, we highlight the importance of focusing on more complex collaborative systems that account for intra-layer collaboration and leverage the full computing continuum. We propose that this shift can be facilitated through the adoption of standardized evaluation environments (discussed in Section 6.3), which simplify the integration and testing of diverse functionalities. For example, mobility could be defined as a configurable agent parameter, while multi-hop offloading could be activated or deactivated via simple interface controls. Similarly, computing layers, the number of devices within each layer, and network connectivity could be easily adjusted to suit different experimental setups. Such a framework would significantly reduce the implementation overhead currently associated with modeling complex systems, thereby enabling researchers to easily incorporate and evaluate multiple dynamic features without the need to construct custom simulations from scratch. This practical step would lower the barrier to studying realistic scenarios and accelerate the development of more robust, generalizable, and deployment-ready offloading strategies.

6.3. Evaluation methodology

The evaluation methodology presents a significant challenge in the field, as there is no standardized benchmarking system for cross-comparing methods. Most studies develop unique simulation setups and define custom metrics, often relying on synthetic tasks. This fragmentation hinders comparative analysis, as authors must assess their methods with tailored setups, limiting cross-validation of results. Consequently, many studies compare their methods only against simplified baselines, such as local computation or single-layer offloading.

While many studies utilize realistic mobility datasets, communication models, and task models, the simulation environments still fail to capture the full complexity of real-world conditions. Real-world environments typically involve a wide variety of obstacles that impact network conditions. In VEC networks, for instance, buildings, other vehicles, and vegetation can affect signal propagation, leading to communication failures between devices. Similarly, in MEC, mobile devices are

often used in crowded areas where various obstacles can interfere with network performance.

To address these challenges, we propose a benchmarking system for the standardized evaluation and comparison of task offloading methods. Such a system would accelerate algorithm development by eliminating the need to recreate simulations from scratch each time. Ideally, the system should be capable of simulating hybrid networks ranging from smaller-scale MEC networks to large-scale VEC, UAV, and satellite networks. It would also provide a platform for cross-validation of approaches, even from different domains, showcasing state-of-the-art methods. While numerical simulation is valuable for evaluating large-scale systems, we believe there is also a need for a physics-based simulator with integrated network modeling. Such a testbed would enable realistic use-case evaluations and support various sensors, including cameras, LiDARs, and radars, which are essential for applications in self-driving, AR/VR, and mapping. Such co-simulation platforms have been proposed in recent years to integrate various simulation tools. For instance, in [151], the authors introduce CAVIAR, a co-simulation platform that combines Airsim for mobility and 3D environments, Sionna for network modeling, and NATS for messaging, with a support for UAVs and cars. Similarly, in [152], the authors propose CAVISE, where mobility and 3D environments are managed by the CARLA simulator, Artery handles communication, OpenCDA is used for self-driving, and SUMO simulates road traffic. A comprehensive survey of Integrated Simulation Environments (ISE) for automated vehicles is provided in [153]. It would enable researchers to evaluate methods using workflows that closely resemble real-world applications, thus accelerating the adoption and deployment of these methods. Additionally, a real-life testbed for multi-user, multi-edge, and cloud environments would further support standardized experiments and facilitate the practical deployment of offloading strategies.

7. Conclusion

This survey presents a comprehensive analysis of computation offloading in collaborative computing systems, examining various methods, architectures, and key offloading dynamics. We classify offloading approaches into classical, heuristic, metaheuristic, and machine learning-based methods, further distinguishing them based on their centralized, decentralized, or distributed implementation. Our findings indicate that classical methods are primarily employed in simpler collaborative systems, whereas heuristic, metaheuristic, and machine learning-based approaches are more suited for complex, multi-layered environments. In particular, machine learning methods are gaining prominence due to their adaptability to complex and dynamic networks. Despite these advancements, several critical challenges remain, particularly in fully leveraging the computing continuum while incorporating intra-layer dynamics such as mobility, multi-hop offloading, and task dependencies. Additionally, the absence of a standardized benchmarking framework hinders the cross-validation of state-of-the-art methods, limiting comparability and reproducibility across studies. Future research should prioritize unifying offloading research across different networks to enhance interoperability, developing benchmarking systems alongside physics-based simulation environments for more realistic evaluations, and advancing offloading approaches for complex collaborative systems and hybrid networks.

CRedit authorship contribution statement

Jussi Kalliola: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Jason J. Jung:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization; **David Hästbacka:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the research project Daze funded by Business Finland as well as the AeroPolis project with funding from the European Union - NextGenerationEU instrument and funded by the Research Council of Finland [grant number 348481].

References

- [1] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, Y. Zhang, Mobile edge cloud system: architectures, challenges, and approaches, *IEEE Syst. J.* 12 (3) (2018) 2495–2508. <https://doi.org/10.1109/JSYST.2017.2654119>
- [2] A. Koubaa, A. Ammar, M. Alahdab, A. Kanhouh, A.T. Azar, DeepBrain: experimental evaluation of cloud-based computation offloading and edge computing in the internet-of-drones for deep learning applications, *Sensors* 20 (18) (2020). <https://doi.org/10.3390/s20185240>
- [3] M. Ahmed, S. Raza, M.A. Mirza, A. Aziz, M.A. Khan, W.U. Khan, J. Li, Z. Han, A survey on vehicular task offloading: classification, issues, and challenges, *J. King Saud Univ. - Comput. Inf. Sci.* 34 (7) (2022) 4135–4162. <https://doi.org/10.1016/j.jksuci.2022.05.016>
- [4] R. Chaâri, O. Cheikhrouhou, A. Koubâa, H. Youssef, T.N. Gia, Dynamic computation offloading for ground and flying robots: taxonomy, state of art, and future directions, *Comput. Sci. Rev.* 45 (2022) 100488. <https://doi.org/10.1016/j.cosrev.2022.100488>
- [5] M.Y. Akhlaqi, Z.B. Mohd Hanapi, Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions, *J. Netw. Comput. Appl.* 212 (2023) 103568.
- [6] K. Sadatdiynov, L. Cui, L. Zhang, J.Z. Huang, S. Salloum, M.S. Mahmud, A review of optimization methods for computation offloading in edge computing networks, *Digit. Commun. Netw.* 9 (2) (2023) 450–461. <https://doi.org/10.1016/j.dcan.2022.03.003>
- [7] H. Long, C. Xu, G. Zheng, Y. Sheng, Socially-aware energy-efficient task partial offloading in MEC networks with D2D collaboration, *IEEE Trans. Green Commun. Netw.* 6 (3) (2022) 1889–1902. <https://doi.org/10.1109/TGCN.2022.3153956>
- [8] C. Qian, G. Zhao, H. Luo, Game theory based D2D collaborative offloading for workflow applications in mobile edge computing, in: 2022 IEEE International Conference on Web Services (ICWS), 2022, pp. 276–285. <https://doi.org/10.1109/ICWS55610.2022.00049>
- [9] Z. Zeng, J. Huang, J. Wu, J. Wu, Joint request offloading and resource allocation for energy efficient D2D enabled multi-type inference services, in: 2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2023, pp. 213–220. <https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom59178.2023.00061>
- [10] X. An, Y. Li, Y. Chen, T. Li, Joint task offloading and resource allocation for multi-user collaborative mobile edge computing, *Comput. Netw.* 250 (2024) 110604.
- [11] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: a survey, *IEEE Internet Things J.* 5 (1) (2018) 450–465. <https://doi.org/10.1109/JIOT.2017.2750180>
- [12] R. Meneguet, R. De Grande, J. Ueyama, G.P.R. Filho, E. Madeira, Vehicular edge computing: architecture, resource management, security, and challenges, *ACM Comput. Surv.* 55 (1) (2021). <https://doi.org/10.1145/3485129>
- [13] S.M.A. Huda, S. Moh, Survey on computation offloading in UAV-enabled mobile edge computing, *J. Netw. Comput. Appl.* 201 (2022) 103341. <https://doi.org/10.1016/j.jnca.2022.103341>
- [14] Y. Lin, W. Feng, Y. Wang, Y. Chen, Y. Zhu, X. Zhang, N. Ge, Y. Gao, Satellite-MEC integration for 6G internet of things: minimal structures, advances, and prospects, *IEEE Open J. Commun. Soc.* 5 (2024) 3886–3903. <https://doi.org/10.1109/OJCOMS.2024.3418860>
- [15] I. Ahammad, Fog computing complete review: concepts, trends, architectures, technologies, simulators, security issues, applications, and open research fields, *SN Comput. Sci.* 4 (6) (2023) 765. <https://doi.org/10.1007/s42979-023-02235-9>
- [16] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, D. Taibi, Cloud continuum: the definition, *IEEE Access* 10 (2022) 131876–131886. <https://doi.org/10.1109/ACCESS.2022.3229185>
- [17] S. Moreschini, F. Pecorelli, X. Li, S. Naz, M. Albano, D. Hästbacka, D. Taibi, Cognitive cloud: the definition, in: 2022 19th International Conference on Distributed Computing and Artificial Intelligence (DCAI), 2023, pp. 219–229.
- [18] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, I. Humar, A dynamic service migration mechanism in edge cognitive computing, *ACM Trans. Internet Technol.* 19 (2) (2019). <https://doi.org/10.1145/3239565>

- [19] S.K.u. Zaman, A.I. Jehangiri, T. Maqsood, Z. Ahmad, A.I. Umar, J. Shuja, E. Alanazi, W. Alasmary, Mobility-aware computational offloading in mobile edge networks: a survey, *Clust. Comput.* 24 (4) (2021) 2735–2756. <https://doi.org/10.1007/s10586-021-03268-6>
- [20] W. Zhang, H. Flores, P. Hui, Towards collaborative multi-device computing, in: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2018, pp. 22–27. <https://doi.org/10.1109/PERCOMW.2018.8480262>
- [21] C. Jiang, X. Cheng, H. Gao, X. Zhou, J. Wan, Toward computation offloading in edge computing: a survey, *IEEE Access* 7 (2019) 131543–131558. <https://doi.org/10.1109/ACCESS.2019.2938660>
- [22] H. Lin, S. Zeadally, Z. Chen, H. Labiod, L. Wang, A survey on computation offloading modeling for edge computing, *J. Netw. Comput. Appl.* 169 (2020) 102781. <https://doi.org/10.1016/j.jnca.2020.102781>
- [23] A. Shakarami, A. Shahidinejad, M. Ghoabei-Arani, A review on the computation offloading approaches in mobile edge computing: a game-theoretic perspective, *Softw. Pract. Exp.* 50 (9) (2020) 1719–1759. <https://doi.org/10.1002/spe.2839>
- [24] B. Wang, C. Wang, W. Huang, Y. Song, X. Qin, A survey and taxonomy on task offloading for edge-cloud computing, *IEEE Access* 8 (2020) 186080–186101. <https://doi.org/10.1109/ACCESS.2020.3029649>
- [25] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, L. Guo, Computation offloading in mobile edge computing networks: a survey, *J. Netw. Comput. Appl.* 202 (2022) 103366. <https://doi.org/10.1016/j.jnca.2022.103366>
- [26] E. Mustafa, J. Shuja, S.K. uz Zaman, A.I. Jehangiri, S. Din, F. Rehman, S. Mustafa, T. Maqsood, A.N. Khan, Joint wireless power transfer and task offloading in mobile edge computing: a survey, *Clust. Comput.* 25 (4) (2022) 2429–2448.
- [27] H. Jin, M.A. Gregory, S. Li, A review of intelligent computation offloading in multi-access edge computing, *IEEE Access* 10 (2022) 71481–71495. <https://doi.org/10.1109/ACCESS.2022.3187701>
- [28] P. Wei, K. Guo, Y. Li, J. Wang, W. Feng, S. Jin, N. Ge, Y.-C. Liang, Reinforcement learning-empowered mobile edge computing for 6G edge intelligence, *IEEE Access* 10 (2022) 65156–65192. <https://doi.org/10.1109/ACCESS.2022.3183647>
- [29] P. Peng, W. Lin, W. Wu, H. Zhang, S. Peng, Q. Wu, K. Li, A survey on computation offloading in edge systems: from the perspective of deep reinforcement learning approaches, *Comput. Sci. Rev.* 53 (2024) 100656. <https://doi.org/10.1016/j.cosrev.2024.100656>
- [30] H. Gu, L. Zhao, Z. Han, G. Zheng, S. Song, AI-enhanced cloud-edge-terminal collaborative network: survey, applications, and future directions, *IEEE Commun. Surv. Tutorials* 26 (2) (2024) 1322–1385. <https://doi.org/10.1109/COMST.2023.3338153>
- [31] S. Zhang, N. Yi, Y. Ma, A survey of computation offloading with task types, *Trans. Intell. Transp. Syst.* 25 (8) (2024) 8313–8333. <https://doi.org/10.1109/TITS.2024.3410896>
- [32] S. Dong, J. Tang, K. Abbas, R. Hou, J. Kamruzzaman, L. Rutkowski, R. Buyya, Task offloading strategies for mobile edge computing: a survey, *Comput. Netw.* 254 (2024) 110791. <https://doi.org/10.1016/j.comnet.2024.110791>
- [33] H. Pirhosseini, E. Zarei, M.H. Rezvani, A. Pourkabirian, A. Shankar, N. Bacanin, C. Maple, W. Viriyasitavat, Game-theoretic methods for edge/fog/cloud computation offloading: a systematic review, *Computing* 107 (8) (2025) 166. <https://doi.org/10.1007/s00607-025-01515-x>
- [34] A. Nabi, S. Moh, Offloading decision and resource allocation in aerial computing: a comprehensive survey, *Comput. Sci. Rev.* 56 (2025) 100734. <https://doi.org/10.1016/j.cosrev.2025.100734>
- [35] J. Pournazari, A. Ullah, A. Al-Dubai, X. Liu, Computation offloading in the edge-to-cloud compute continuum: a survey of federated architectural solutions, *Clust. Comput.* 28 (13) (2025) 839. <https://doi.org/10.1007/s10586-025-05577-6>
- [36] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges, *IEEE Internet Things J.* 3 (5) (2016) 637–646. <https://doi.org/10.1109/JIOT.2016.2579198>
- [37] A. Al-Dulaimy, M. Jansen, B. Johansson, A. Trivedi, A. Iosup, M. Ashjaei, A. Galletta, D. Kimovski, R. Prodan, K. Tserpes, G. Kousiouris, C. Giannakos, I. Brandic, N. Ali, A.B. Bondi, A.V. Papadopoulos, The computing continuum: from IoT to the cloud, *Internet Things* 27 (2024) 101272. <https://doi.org/10.1016/j.iot.2024.101272>
- [38] J. Feng, L. Zhao, J. Du, X. Chu, F.R. Yu, Computation offloading and resource allocation in D2D-enabled mobile edge computing, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6. <https://doi.org/10.1109/ICC.2018.8422776>
- [39] H. Zhang, Y. Li, X. Zhang, Joint D2D and base station collaboration with hierarchical task offloading in MEC networks, in: 2024 IEEE/CIC International Conference on Communications in China (ICCC), 2024, pp. 491–496. <https://doi.org/10.1109/ICCC62479.2024.10681930>
- [40] J. Wu, M. Tang, C. Jiang, L. Gao, B. Cao, Cloud-edge-end collaborative task offloading in vehicular edge networks: a multilayer deep reinforcement learning approach, *IEEE Internet Things J.* 11 (22) (2024) 36272–36290. <https://doi.org/10.1109/JIOT.2024.3472472>
- [41] W. Fan, Hybrid deep reinforcement learning-based task offloading for D2D-assisted cloud-edge-device collaborative networks, *IEEE Trans. Mob. Comput.* 23 (12) (2024) 13455–13471.
- [42] C. Zhang, C. Wu, M. Lin, Y. Lin, W. Liu, Proximal policy optimization for efficient D2D-assisted computation offloading and resource allocation in multi-access edge computing, *Future Internet* 16 (1) (2024). <https://doi.org/10.3390/fi16010019>
- [43] P. Hou, X. Jiang, Z. Lu, B. Li, Z. Wang, Joint computation offloading and resource allocation based on deep reinforcement learning in C-V2X edge computing, *Appl. Intell.* 53 (19) (2023) 22446–22466. <https://doi.org/10.1007/s10489-023-04637-x>
- [44] E. Wang, H. Wang, P.-M. Dong, Y.-B. Xu, Y.-J. Yang, Distributed game-theoretical D2D-enabled task offloading in mobile edge computing, *J. Comput. Sci. Technol.* 37 (4) (2022) 919–941. <https://doi.org/10.1007/s11390-022-2063-3>
- [45] Q. Huo, W. Zhang, Z. Wu, G. Song, B. Wang, Collaborative offloading strategy for dependent tasks in mobile edge computing, *Wirel. Pers. Commun.* 134 (1) (2024) 267–292. <https://doi.org/10.1007/s11277-024-10904-y>
- [46] J. Lu, W. Feng, D. Pu, Resource allocation and offloading decisions of D2D collaborative UAV-assisted MEC systems, *KSII Trans. Internet Inf. Syst.* 18 (1) (2024) 211–232. <https://doi.org/10.3837/tiis.2024.01.012>
- [47] D. Cao, B. Peng, Y. Wang, F. Alqahtani, J. Zhang, J. Wang, Dual-timescale resource management for multi-type caching placement and multi-user computation offloading in internet of vehicle, *Simul. Model. Pract. Theory* 138 (2025) 103025.
- [48] X. Wang, J. Ye, J.C.S. Lui, Joint D2D collaboration and task offloading for edge computing: a mean field graph approach, in: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), 2021, pp. 1–10. <https://doi.org/10.1109/IWQOS52092.2021.9521271>
- [49] J. Xue, X. Guan, Collaborative computation offloading and resource allocation based on dynamic pricing in mobile edge computing, *Comput. Commun.* 198 (2023) 52–62.
- [50] W. Sun, H. Zhang, L. Wang, S. Guo, D. Yuan, Profit maximization task offloading mechanism with D2D collaboration in MEC networks, in: 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), 2019, pp. 1–6. <https://doi.org/10.1109/WCSP.2019.8928117>
- [51] N. Wan, Y. Luo, G. Zeng, X. Zhou, Minimization of VANET execution time based on joint task offloading and resource allocation, *Peer-to-Peer Netw. Appl.* 16 (1) (2023) 71–86. <https://doi.org/10.1007/s12083-022-01385-6>
- [52] S. Shao, L. Su, Q. Zhang, S. Wu, S. Guo, F. Qi, Multi task dynamic edge-end computing collaboration for urban internet of vehicles, *Comput. Netw.* 227 (2023) 109690. <https://doi.org/10.1016/j.comnet.2023.109690>
- [53] Y. Lu, D. Han, X. Wang, Q. Gao, Enhancing vehicular edge computing system through cooperative computation offloading, *Clust. Comput.* 26 (1) (2022) 771–788. <https://doi.org/10.1007/s10586-022-03803-z>
- [54] K. Li, X. Wang, Q. He, M. Yang, M. Huang, S. Dustdar, Task computation offloading for multi-access edge computing via attention communication deep reinforcement learning, *IEEE Trans. Serv. Comput.* 16 (4) (2023) 2985–2999. <https://doi.org/10.1109/TSC.2022.3225473>
- [55] G. Qiao, S. Leng, Y. Zhang, Online learning and optimization for computation offloading in D2D edge computing and networks, *Mob. Netw. Appl.* 27 (3) (2022) 1111–1122. <https://doi.org/10.1007/s11036-018-1176-y>
- [56] B. Lin, K. Lin, C. Lin, Y. Lu, Z. Huang, X. Chen, Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing, *J. Cloud Comput.* 10 (1) (2021) 33. <https://doi.org/10.1186/s13677-021-00246-6>
- [57] P. Qin, Y. Wang, Z. Cai, J. Liu, J. Li, X. Zhao, MADRL-based URLLC-aware task offloading for air-ground vehicular cooperative computing network, *IEEE Trans. Intell. Transp. Syst.* 25 (7) (2024) 6716–6729. <https://doi.org/10.1109/TITS.2023.3342271>
- [58] A.D. Wyner, Recent results in the Shannon theory, *IEEE Trans. Inf. Theory* 20 (1) (1974) 2–10. <https://doi.org/10.1109/TIT.1974.1055171>
- [59] C. Liu, K. Liu, H. Ren, X. Xu, R. Xie, J. Cao, RTDS: real-time distributed strategy for multi-period task offloading in vehicular edge computing environment, *Neural Comput. Appl.* 35 (17) (2021) 12373–12387. <https://doi.org/10.1007/s00521-021-05766-5>
- [60] T.N. Dang, A. Manzoor, Y.K. Tun, S.M.A. Kazmi, R. Haw, S.H. Hong, Z. Han, C.S. Hong, Joint communication, computation, and control for computational task offloading in vehicle-assisted multi-access edge computing, *IEEE Access* 10 (2022) 122513–122529. <https://doi.org/10.1109/ACCESS.2022.3220251>
- [61] T. Li, L. Zhao, C. Mao, N. Lin, Q. He, X. Wang, A genetic algorithm based adaptive offloading scheme for domain generalization in VEC, in: 2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2023, pp. 161–168. <https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom59178.2023.00054>
- [62] Z. Song, R. Ma, Y. Xie, A collaborative task offloading strategy for mobile edge computing in internet of vehicles, in: 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2021, pp. 1379–1384. <https://doi.org/10.1109/IAEAC50856.2021.9390817>
- [63] W. Gao, X. Yang, A joint resource allocation and task offloading scheme for energy-aware and latency constrained vehicular edge computing network, in: 2024 IEEE/CIC International Conference on Communications in China (ICCC), 2024, pp. 167–172. <https://doi.org/10.1109/ICCC62479.2024.10681851>
- [64] T. Deng, Y. Zhao, X. Zhang, B. Xu, J. Qi, Y. Ma, A mobile device multitasking model with multiple mobile edge computing servers, in: 2020 6th International Conference on Artificial Intelligence and Security (ICAIS), 2020, pp. 151–161. https://doi.org/10.1007/978-3-030-57884-8_14
- [65] Y. Cui, L. Du, H. Wang, D. Wu, R. Wang, Reinforcement learning for joint optimization of communication and computation in vehicular networks, *IEEE Trans. Veh. Technol.* 70 (12) (2021) 13062–13072. <https://doi.org/10.1109/TVT.2021.3125109>
- [66] J. Xue, F. Shao, T. Zhang, G. Tian, H. Jiang, VEC collaborative task offloading and resource allocation based on deep reinforcement learning under parking assistance, *Wirel. Pers. Commun.* 136 (1) (2024) 321–345. <https://doi.org/10.1007/s11277-024-11258-1>
- [67] R. Ranji, A.M. Mansoor, A.A. Sani, EEDOS: an energy-efficient and delay-aware offloading scheme based on device to device collaboration in mobile edge com-

- puting, Telecommun. Syst. 73 (2) (2020) 171–182. <https://doi.org/10.1007/s11235-019-00595-3>
- [68] L. Wu, R. Zhang, Q. Li, C. Ma, X. Shi, A mobile edge computing-based applications execution framework for internet of vehicles, Front. Comput. Sci. 16 (5) (2022). <https://doi.org/10.1007/s11704-021-0425-6>
- [69] S. Yu, B. Dab, Z. Movahedi, R. Langar, L. Wang, A socially-aware hybrid computation offloading framework for multi-access edge computing, IEEE Trans. Mob. Comput. 19 (6) (2020) 1247–1259. <https://doi.org/10.1109/TMC.2019.2908154>
- [70] C. Wang, X. Chai, S. Peng, Y. Yuan, G. Li, Deep reinforcement learning with entropy and attention mechanism for D2D-assisted task offloading in edge computing, IEEE Trans. Serv. Comput. 17 (6) (2024) 3317–3329. <https://doi.org/10.1109/TSC.2024.3495503>
- [71] C. Wang, Z. Wang, W. Guan, W. Wang, L. Xu, L. Li, S. Huang, W. Wang, Trustworthy health monitoring based on distributed wearable electronics with edge intelligence, IEEE Trans. Consum. Electron. 70 (1) (2024) 2333–2341. <https://doi.org/10.1109/TCE.2024.3358803>
- [72] H. Hao, C. Xu, W. Zhang, S. Yang, G.-M. Muntean, Joint task offloading, resource allocation, and trajectory design for multi-UAV cooperative edge computing with task priority, IEEE Trans. Mob. Comput. 23 (9) (2024) 8649–8663. <https://doi.org/10.1109/TMC.2024.3350078>
- [73] M.I. Khaleel, M. Safran, S. Alfarhood, D. Gupta, Combinatorial metaheuristic methods to optimize the scheduling of scientific workflows in green DVFS-enabled edge-cloud computing, Alex. Eng. J. 86 (2024) 458–470. <https://doi.org/10.1016/j.aej.2023.11.074>
- [74] M.I. Khaleel, M. Safran, S. Alfarhood, M. Zhu, Workflow scheduling scheme for optimized reliability and end-to-end delay control in cloud computing using AI-based modeling, Mathematics 11 (20) (2023). <https://doi.org/10.3390/math11204334>
- [75] J. Tang, W. Zhao, J. Jin, Y. Xiang, X. Wang, Z. Zhou, Adaptive search and collaborative offloading under device-to-device joint edge computing network, IEEE Trans. Mob. Comput. 24 (10) (2025) 9852–9867. <https://doi.org/10.1109/TMC.2025.3567549>
- [76] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wirel. Commun. Mob. Comput. 2 (2002). <https://doi.org/10.1002/wcm.72>
- [77] A. Chopra, A.U. Rahman, A.W. Malik, S.D. Ravana, Adaptive-learning-based vehicle-to-vehicle opportunistic resource-sharing framework, IEEE Internet Things J. 9 (14) (2022) 12497–12504. <https://doi.org/10.1109/JIOT.2021.3137264>
- [78] Q. Luo, C. Li, T.H. Luan, W. Shi, Collaborative data scheduling for vehicular edge computing via deep reinforcement learning, IEEE Internet Things J. 7 (10) (2020) 9637–9650. <https://doi.org/10.1109/JIOT.2020.2983660>
- [79] M. Luca, G. Barlacchi, B. Lepri, L. Pappalardo, A survey on deep learning for human mobility, ACM Comput. Surv. 55 (1) (2021). <https://doi.org/10.1145/3485125>
- [80] F. Ekman, A. Keränen, J. Karvo, J. Ott, Working day movement model, in: 2008 Proceedings of the 1st ACM SIGMOBILE Workshop on Mobility Models (Mobility-Models'08), 2008, pp. 33–40. <https://doi.org/10.1145/1374688.1374695>
- [81] X. Chen, L. Pu, L. Gao, W. Wu, D. Wu, Exploiting massive D2D collaboration for energy-efficient mobile edge computing, IEEE Wirel. Commun. 24 (4) (2017) 64–71. <https://doi.org/10.1109/MWC.2017.1600321>
- [82] S. Zhao, Y. Yang, X. Yang, W. Zhang, X. Luo, H. Qian, Online user association and computation offloading for fog-enabled D2D network, in: 2017 IEEE Fog World Congress (FWC), 2017, pp. 1–6. <https://doi.org/10.1109/FWC.2017.8368534>
- [83] M. Dai, S. Xu, Z. Wang, H. Ma, X. Qiu, Edge trusted sharing: task-driven decentralized resources collaborate in IoT, IEEE Internet Things J. 10 (14) (2023) 12077–12089. <https://doi.org/10.1109/JIOT.2021.3123333>
- [84] Y. Lan, X. Wang, C. Wang, D. Wang, Q. Li, Collaborative computation offloading and resource allocation in cache-aided hierarchical edge-cloud systems, Electronics 8 (12) (2019). <https://doi.org/10.3390/electronics8121430>
- [85] Z. Li, X. Jiang, D. Wu, H. Wang, T. Tang, R. Wang, Video service-oriented vehicular collaboration: a multi-agent proximal policy optimization approach, in: 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6. <https://doi.org/10.1109/GLOBECOM46510.2021.9685857>
- [86] Y. Zhang, M. Zhang, C. Fan, F. Li, B. Li, Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment, EURASIP J. Adv. Signal Process. 2021 (1) (2021) 33. <https://doi.org/10.1186/s13634-021-00750-6>
- [87] Z. Wei, R. He, H. Liu, C. Song, Joint computation offloading and resource allocation in green MEC-assisted software-defined island internet of things, IEEE Internet Things J. 12 (1) (2025) 140–162. <https://doi.org/10.1109/JIOT.2024.3459098>
- [88] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, M.-T. Zhou, MEETS: maximal energy efficient task scheduling in homogeneous fog networks, IEEE Internet Things J. 5 (5) (2018) 4076–4087. <https://doi.org/10.1109/JIOT.2018.2846644>
- [89] X. Han, D. Tian, Z. Sheng, X. Duan, J. Zhou, W. Hao, K. Long, M. Chen, V.C.M. Leung, Reliability-aware joint optimization for cooperative vehicular communication and computing, IEEE Trans. Intell. Transp. Syst. 22 (8) (2021) 5437–5446. <https://doi.org/10.1109/TITS.2020.3038558>
- [90] X. Chen, Z. Zhou, W. Wu, D. Wu, J. Zhang, Socially-motivated cooperative mobile edge computing, IEEE Netw. 32 (6) (2018) 177–183. <https://doi.org/10.1109/MNET.2018.1700354>
- [91] D. Wu, F. Wang, X. Cao, J. Xu, Joint communication and computation optimization for wireless powered mobile edge computing with D2D offloading, J. Commun. Inf. Netw. 4 (4) (2019) 72–86.
- [92] S. Raza, W. Liu, M. Ahmed, M.R. Anwar, M.A. Mirza, Q. Sun, S. Wang, An efficient task offloading scheme in vehicular edge computing, J. Cloud Comput. 9 (1) (2020). <https://doi.org/10.1186/s13677-020-00175-w>
- [93] B. Xu, T. Deng, Y. Liu, Y. Zhao, Z. Xu, J. Qi, S. Wang, D. Liu, Optimization of cooperative offloading model with cost consideration in mobile edge computing, Soft Comput. 27 (12) (2022) 8233–8243. <https://doi.org/10.1007/s00500-022-07733-1>
- [94] M. Liu, F.R. Yu, Y. Teng, V.C.M. Leung, M. Song, Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, IEEE Trans. Wirel. Commun. 18 (1) (2019) 695–708. <https://doi.org/10.1109/TWC.2018.2885266>
- [95] H. Liu, Z. Niu, J. Du, X. Lin, Genetic algorithm for delay efficient computation offloading in dispersed computing, Ad Hoc Netw. 142 (2023) 103109.
- [96] J. Jiang, K. Su, Z. Lu, Fully decentralized cooperative multi-agent reinforcement learning: a survey, 2024, arXiv:2401.04934
- [97] R. Li, T. Ouyang, L. Zeng, G. Liao, Z. Zhou, X. Chen, Online optimization of DNN inference network utility in collaborative edge computing, IEEE/ACM Trans. Netw. 32 (5) (2024) 4414–4426. <https://doi.org/10.1109/TNET.2024.3421356>
- [98] C. Li, S. Liu, K. Jiang, M. Yang, Z. Zhang, B. Wang, L. Zhao, C. Chen, S. Wan, DNN inference acceleration based on adaptive task partitioning and offloading in embedded VEC, ACM Trans. Embed. Comput. Syst. 24 (4) (2025). <https://doi.org/10.1145/3725734>
- [99] D. Yang, C. Zhan, Y. Yang, H. Yan, W. Liu, Integrating UAVs and D2D communication for MEC network: a collaborative approach to caching and computation, IEEE Trans. Technol. 74 (6) (2025) 10041–10046. <https://doi.org/10.1109/TVT.2025.3540915>
- [100] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2016) 2795–2808. <https://doi.org/10.1109/TNET.2015.2487344>
- [101] Y. Hao, M. Chen, L. Hu, M.S. Hossain, A. Ghoneim, Energy efficient task caching and offloading for mobile edge computing, IEEE Access 6 (2018) 11365–11373. <https://doi.org/10.1109/ACCESS.2018.2805798>
- [102] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, Y. Zhang, Selective offloading in mobile edge computing for the green internet of things, IEEE Netw. 32 (1) (2018) 54–60. <https://doi.org/10.1109/MNET.2018.1700101>
- [103] X. Chen, J. Zhang, When D2D meets cloud: hybrid mobile task offloadings in fog computing, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–6. <https://doi.org/10.1109/ICC.2017.7996590>
- [104] H. Li, H. Xu, C. Zhou, X. Lü, Z. Han, Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment, IEEE Trans. Veh. Technol. 69 (9) (2020) 10214–10226. <https://doi.org/10.1109/TVT.2020.3003898>
- [105] X. Fu, B. Tang, F. Guo, L. Kang, Priority and dependency-based DAG tasks offloading in fog/edge collaborative environment, in: 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 440–445. <https://doi.org/10.1109/CSCWD49262.2021.9437784>
- [106] Y. Han, Z. Zhao, J. Mo, C. Shu, G. Min, Efficient task offloading with dependency guarantees in ultra-dense edge networks, in: 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1–6. <https://doi.org/10.1109/GLOBECOM38437.2019.9013142>
- [107] A. Bozorgchenani, D. Tarchi, G.E. Corazza, Mobile edge computing partial offloading techniques for mobile urban scenarios, in: 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1–6. <https://doi.org/10.1109/GLOCOM.2018.8647240>
- [108] J. Zhao, A.M. El-Sherbeen, Joint task offloading and multi-task offloading based on NOMA enhanced internet of vehicles in edge computing, J. Grid Comput. 22 (1) (2024) 28. <https://doi.org/10.1007/s10723-024-09748-4>
- [109] P.-Q. Huang, Y. Wang, K. Wang, Z.-Z. Liu, A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing, IEEE Trans. Cybern. 50 (10) (2020) 4228–4241. <https://doi.org/10.1109/TCYB.2019.2916728>
- [110] F. Zhang, M.M. Wang, Stochastic congestion game for load balancing in mobile-edge computing, IEEE Internet Things J. 8 (2) (2021) 778–790. <https://doi.org/10.1109/JIOT.2020.3008009>
- [111] T. Fang, F. Yuan, L. Ao, J. Chen, Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: a potential game approach, IEEE Internet Things J. 9 (5) (2022) 3226–3237. <https://doi.org/10.1109/JIOT.2021.3097754>
- [112] C. Qian, G. Zhao, H. Luo, A novel traversal search-based D2D collaborative offloading approach for workflow application in dynamic edge environment, in: 2022 Computer Supported Cooperative Work and Social Computing (ChineseCSCW 2021), 2022, pp. 176–190.
- [113] M.A. Rodriguez, R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, IEEE Trans. Cloud Comput. 2 (2) (2014) 222–235. <https://doi.org/10.1109/TCC.2014.2314655>
- [114] E. Wang, D. Luan, Y. Yang, Z. Wang, P. Dong, D. Li, W. Liu, J. Wu, Distributed game-theoretical route navigation for vehicular crowdsensing, in: 2021 Proceedings of the 50th International Conference on Parallel Processing (ICPP), 2021, pp. 1–11. <https://doi.org/10.1145/3472456.3472498>
- [115] K. Cao, J. Weng, K. Li, Reliability-driven end-end-edge collaboration for energy minimization in large-scale cyber-physical systems, IEEE Trans. Reliab. 73 (1) (2024) 230–244. <https://doi.org/10.1109/TR.2023.3297124>
- [116] L. Pu, X. Chen, J. Xu, X. Fu, D2D fogging: an energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration, IEEE J. Sel. Areas Commun. 34 (12) (2016) 3887–3901. <https://doi.org/10.1109/JSAC.2016.2624118>
- [117] X. Cao, F. Wang, J. Xu, R. Zhang, S. Cui, Joint computation and communication cooperation for energy-efficient mobile edge computing, IEEE Internet Things J. 6 (3) (2019) 4188–4200. <https://doi.org/10.1109/JIOT.2018.2875246>
- [118] Y. Yang, C. Long, J. Wu, S. Peng, B. Li, D2D-enabled mobile-edge computation offloading for multiuser IoT network, IEEE Internet Things J. 8 (16) (2021) 12490–12504. <https://doi.org/10.1109/JIOT.2021.3068722>

- [119] B. Hu, Z. Cao, M. Zhou, Energy-minimized scheduling of real-time parallel workflows on heterogeneous distributed computing systems, *IEEE Trans. Serv. Comput.* 15 (5) (2022) 2766–2779. <https://doi.org/10.1109/TSC.2021.3054754>
- [120] W. Cheng, W. Wei, S. Zhang, Dynamic willingness-driven collaborative task offloading and resource optimization in D2D-assisted MEC for IIoT, in: 2025 IEEE/CIC International Conference on Communications in China (ICCC), 2025, pp. 1–6. <https://doi.org/10.1109/ICCC65529.2025.11149263>
- [121] S. Li, W. Li, W. Zheng, Y. Xia, K. Guo, Q. Peng, X. Li, J. Ren, Multi-user joint task offloading and resource allocation based on mobile edge computing in mining scenarios, *Sci. Rep.* 15 (1) (2025) 16170. <https://doi.org/10.1038/s41598-025-00730-y>
- [122] Y. Yu, J. Zhang, K.B. Letaief, Joint subcarrier and CPU time allocation for mobile edge computing, in: 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6. <https://doi.org/10.1109/GLOCOM.2016.7841937>
- [123] Y. Wang, P. Lang, D. Tian, J. Zhou, X. Duan, Y. Cao, D. Zhao, A game-based computation offloading method in vehicular multiaccess edge computing networks, *IEEE Internet Things J.* 7 (6) (2020) 4987–4996. <https://doi.org/10.1109/JIOT.2020.2972061>
- [124] X. Wang, Z. Ning, S. Guo, L. Wang, Imitation learning enabled task scheduling for online vehicular edge computing, *IEEE Trans. Mob. Comput.* 21 (2) (2022) 598–611. <https://doi.org/10.1109/TMC.2020.3012509>
- [125] Q. Shen, B.-J. Hu, E. Xia, Cuckoo search-enabled task scheduling and cache updating in vehicular edge-fog computing, *IEEE Trans. Veh. Technol.* 74 (6) (2025) 9656–9670. <https://doi.org/10.1109/TVT.2025.3540639>
- [126] A. Virdis, G. Stea, G. Nardini, Simulating LTE/LTE-advanced networks with SimuLTE, in: 2014 4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2014), 2015, pp. 83–105.
- [127] W. Hou, H. Wen, H. Song, W. Lei, W. Zhang, Multiagent deep reinforcement learning for task offloading and resource allocation in cybertwin-based networks, *IEEE Internet Things J.* 8 (22) (2021) 16256–16268. <https://doi.org/10.1109/JIOT.2021.3095677>
- [128] Z. Li, C. Guo, Y. Xuan, A multi-agent deep reinforcement learning based spectrum allocation framework for D2D communications, in: 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1–6. <https://doi.org/10.1109/GLOBECOM38437.2019.9013763>
- [129] H. Ye, G.Y. Li, B.-H.F. Juang, Deep reinforcement learning based resource allocation for V2V communications, *IEEE Trans. Veh. Technol.* 68 (4) (2019) 3163–3173. <https://doi.org/10.1109/TVT.2019.2897134>
- [130] M. Sun, X. Xu, S. Han, H. Zheng, X. Tao, P. Zhang, Secure computation offloading for device-collaborative MEC networks: a DRL-based approach, *IEEE Trans. Veh. Technol.* 72 (4) (2023) 4887–4903. <https://doi.org/10.1109/TVT.2022.3227197>
- [131] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, M. Bennis, Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning, *IEEE Internet Things J.* 6 (3) (2019) 4005–4018. <https://doi.org/10.1109/JIOT.2018.2876279>
- [132] H. Zhang, N. Yang, W. Huangfu, K. Long, V.C.M. Leung, Power control based on deep reinforcement learning for spectrum sharing, *IEEE Trans. Wirel. Commun.* 19 (6) (2020) 4209–4219. <https://doi.org/10.1109/TWC.2020.2981320>
- [133] Z. Tong, H. Chen, X. Deng, K. Li, K. Li, A scheduling scheme in the cloud computing environment using deep Q-learning, *Inf. Sci.* 512 (2020) 1170–1191. <https://doi.org/10.1016/j.ins.2019.10.035>
- [134] S. Wang, S. Zhao, H. Gui, X. He, Z. Lu, B. Chen, Z. Fan, S. Pang, Energy-efficient collaborative task offloading in multi-access edge computing based on deep reinforcement learning, *Ad Hoc Netw.* 169 (2025) 103743. <https://doi.org/10.1016/j.adhoc.2024.103743>
- [135] L. Xu, H. Zhuang, L. Zhang, G. Pang, Adaptive computation offloading based on enhanced multi-agent deep reinforcement learning, in: 2025 IEEE Wireless Communications and Networking Conference (WCNC), 2025, pp. 1–6. <https://doi.org/10.1109/WCNC61545.2025.10978374>
- [136] S. Guo, J. Liu, Y. Yang, B. Xiao, Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing, *IEEE Trans. Mob. Comput.* 18 (2) (2019) 319–333. <https://doi.org/10.1109/TMC.2018.2831230>
- [137] S. Nath, J. Wu, Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems, *Intell. Convergent Netw.* 1 (2) (2020) 181–198. <https://doi.org/10.23919/ICN.2020.0014>
- [138] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, R.Y.K. Kwok, Intelligent edge computing in internet of vehicles: a joint computation offloading and caching solution, *IEEE Trans. Intell. Transp. Syst.* 22 (4) (2021) 2212–2225. <https://doi.org/10.1109/TITS.2020.2997832>
- [139] M. Al-khafaji, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, Y. Jararweh, Improving fog computing performance via fog-2-fog collaboration, *Future Gener. Comput. Syst.* 100 (2019) 266–280. <https://doi.org/10.1016/j.future.2019.05.015>
- [140] H. Liao, Z. Zhou, X. Zhao, B. Ai, S. Mumtaz, Task offloading for vehicular fog computing under information uncertainty: a matching-learning approach, in: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), 2019, pp. 2001–2006. <https://doi.org/10.1109/IWCMC.2019.8766579>
- [141] Y. Ullah, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, Z. Niu, Adaptive learning-based task offloading for vehicular edge computing systems, *IEEE Trans. Veh. Technol.* 68 (4) (2019) 3061–3074. <https://doi.org/10.1109/TVT.2019.2895593>
- [142] C. Wu, Z. Huang, Y. Zou, Delay constrained hybrid task offloading of internet of vehicle: a deep reinforcement learning method, *IEEE Access* 10 (2022) 102778–102788. <https://doi.org/10.1109/ACCESS.2022.3206359>
- [143] M. Liwang, J. Wang, G. Zhibin, X. Du, M. Guizani, Game theory based opportunistic computation offloading in cloud-enabled IoV, *IEEE Access* 7 (2019) 32551–32561. <https://doi.org/10.1109/ACCESS.2019.2897617>
- [144] I. Ullah, Y.-H. Han, Optimizing vehicular edge computing: graph-based double-DQN approaches for intelligent task offloading, *J. Supercomput.* 81 (1) (2024) 76. <https://doi.org/10.1007/s11227-024-06599-4>
- [145] J. Xue, H. Wu, R. Zhang, Z. Wang, Research on collaborative offloading and resource allocation of internet of vehicles in ISAC scenarios based on digital twin, *Ad Hoc Netw.* 179 (2025) 104009. <https://doi.org/10.1016/j.adhoc.2025.104009>
- [146] Y. Xu, Y. Lu, Y. Dai, C. Sun, J. Liu, M. Guizani, H. Wu, Federated graph neural networks for dynamic computation offloading in vehicular networks, in: 2024 IEEE Global Communications Conference (GLOBECOM), 2024, pp. 4196–4201. <https://doi.org/10.1109/GLOBECOM52923.2024.10901545>
- [147] D. Cao, Y. Wang, Y. Yang, S. He, Joint optimization of multi-type caching placement and multi-user computation offloading for vehicular edge computing, in: 2023 IEEE Global Communications Conference (GLOBECOM), 2023, pp. 5593–5598. <https://doi.org/10.1109/GLOBECOM54140.2023.10437983>
- [148] J. Xue, L. Wang, Q. Yu, P. Mao, Multi-agent deep reinforcement learning-based partial offloading and resource allocation in vehicular edge computing networks, *Comput. Commun.* 234 (2025) 108081. <https://doi.org/10.1016/j.comcom.2025.108081>
- [149] X. Chai, X. Li, X. Li, H. Cai, F. Huang, Task-oriented collaborative computing scheme for delay-constrained vehicular networks, *Alex. Eng. J.* 117 (2025) 230–240. <https://doi.org/10.1016/j.aej.2025.01.052>
- [150] X. Huang, R. Zou, W. Zhong, J. Kang, Y. Qi, M. Hao, Deep reinforcement learning for hybrid task scheduling in collaborative vehicular edge computing, in: 2024 20th International Conference on Mobility, Sensing and Networking (MSN), 2024, pp. 353–358. <https://doi.org/10.1109/MSN63567.2024.00056>
- [151] J. Borges, F. Bastos, I. Correa, P. Batista, A. Klautau, CAVIAR: co-simulation of 6G communications, 3-D scenarios, and AI for digital twins, *IEEE Internet Things J.* 11 (19) (2024) 31287–31300. <https://doi.org/10.1109/JIOT.2024.3418675>
- [152] V.G. Stepanyants, A.Y. Romanov, Influence of realistic perception and surroundings on qualitative results in automated and connected vehicle simulation, *IEEE Access* 12 (2024) 43721–43733. <https://doi.org/10.1109/ACCESS.2024.3380369>
- [153] V.G. Stepanyants, A.Y. Romanov, A survey of integrated simulation environments for connected automated vehicles: requirements, tools, and architecture, *IEEE Intell. Transp. Syst. Mag.* 16 (2) (2024) 6–22. <https://doi.org/10.1109/ITS.2023.3335126>

Jussi Kalliola [corresponding author] received a BA degree in Informatics Systems from Vytautas Magnus University, Lithuania, and an MSc degree in Machine Learning and Signal Processing from the University of Tampere, Finland, in 2020 and 2023, respectively. He is currently pursuing a PhD in Computing and Electrical Engineering at the University of Tampere. His research interests include distributed computing, edge computing, robotics, networked robotics, and computer vision.

Jason J. Jung is a Full Professor in Chung-Ang University, Korea, since September 2014. Before joining CAU, he was an Assistant Professor in Yeungnam University, Korea since 2007. Also, he was a postdoctoral researcher in INRIA Rhone-Alpes, France in 2006, and a visiting scientist in Fraunhofer Institute (FIRST) in Berlin, Germany in 2004. He received the BEng in Computer Science and Mechanical Engineering from Inha University in 1999. He received MS and PhD degrees in Computer and Information Engineering from Inha University in 2002 and 2005, respectively. His research topics are knowledge engineering on social networks by using many types of AI methodologies, e.g., data mining, machine learning, and logical reasoning. Recently, he has been working on intelligent schemes to understand various social dynamics in large scale social media.

David Hästbacka is an Associate Professor in Software Engineering at Tampere University, Finland. He received the MSc and DSc degrees in Automation Science and Engineering from Tampere University of Technology, Tampere, Finland, in 2007 and 2013, respectively. His research interests include software systems architectures and data architectures spanning edge and cloud, as well as interoperability in such software systems. Application areas include industrial production, energy systems, autonomous systems, and other smart environments.