Computer Science Review 45 (2022) 100488

Contents lists available at ScienceDirect

Computer Science Review

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

Review article

Dynamic computation offloading for ground and flying robots: Taxonomy, state of art, and future directions

Rihab Chaâri [a,b,*], Omar Cheikhrouhou [c,d], Anis Koubâa [f,e], Habib Youssef [b], Tuan Nguyen Gia [g]

[a] National School of Computer Science and Engineering (ENSI), University of Mannouba, Tunisia
[b] Prince Research Laboratory, University of Sousse, Tunisia
[c] Higher Institute of Computer Science of Mahdia, University of Monastir, Mahdia 5111, Tunisia
[d] CES Laboratory, National School of Engineers of Sfax, University of Sfax, Sfax 3038, Tunisia
[e] Prince Sultan University, Saudi Arabia
[f] CISTER/INESC-TEC, ISEP, Portugal
[g] Department of Computing, University of Turku, Turku, Finland

ARTICLE INFO

Article history:
Received 6 June 2021
Received in revised form 15 May 2022
Accepted 30 May 2022
Available online 8 July 2022

Keywords:
Computation offloading
Offloading decision
Robots
Optimization

ABSTRACT

The collaboration between ground and flying robots can take advantage of the capabilities of each system to enhance the performance of robotic applications, including military, healthcare, disaster management, etc. Unfortunately, this cooperation is restricted by the physical constraint that all computation should be performed on robots. The concept of computation offloading has great potential to improve the performance of both ground and flying robots. Unfortunately, external conditions like the network conditions, the robot's mobility, and the availability of the processing resources may lead to new challenges. Accordingly, these challenges should be addressed from different perspectives, like security, network communication, response time, and energy consumption. Recently, most computation offloading solutions aim to optimize further the robot's energy consumption. Several research works are designed 1.) to satisfy the real-time requirements of robotic applications and 2.) to solve the trade-off between the energy consumed by computation and communication. To better understand these concepts, we present a comprehensive overview of the computation offloading process for ground and flying robots. We also devised a taxonomy explaining the factors affecting the offloading decision in a robotic scenario. The taxonomy presents guidelines to recognize the scope of research in offloading decisions that were designed for robots. Then, we discuss the state-of-the-art techniques of computation offloading from an architectural point of view, and we survey works related to offloading decisions for robots.

Contents

* Corresponding author at: National School of Computer Science and Engineering (ENSI), University of Mannouba, Tunisia.
E-mail addresses: rihab.chaari@coins-lab.org (R. Chaâri), omar.cheikhrouhou@isetsf.rnu.tn (O. Cheikhrouhou), akoubaa@psu.edu.sa (A. Koubâa), Habib.Youssef@fsm.rnu.tn (H. Youssef), tunggi@utu.fi (T.N. Gia).

https://doi.org/10.1016/j.cosrev.2022.100488
1574-0137/

1. Introduction

The collaboration between ground and flying robots has significantly improved the execution of many applications since it offers the advantages of heterogeneous robotic platforms [1]. For instance, fire management may require the employment of (1.) UAVs for fire detection because ground robots cannot traverse all the land, and (2.) ground robots are usually needed in the extinguishing phase. In search and rescue scenarios, the collaboration of aerial and ground robots has been proved to be more efficient than a single type robot system [2]. In these applications, UAVs help ground robots to find paths to target victims based on their spacial localization; and ground robots are responsible for rescuing these targets on the ground [3]. Air–ground robots can also be used in agriculture applications [4] to improve agriculture management, for industrial environment perception [5], etc. In fact, ground robots, like humanoid robots, are designed to interact with their surrounding environment and can transport payloads. A flying robot can produce a positional assessment of the environment because of its ability to consistently cover huge regions with its bird's-eye view [6]. A flying robot refers to Unmanned Aerial Vehicles (UAVs), drones, or airships [7].

The advancements in wireless communication technologies promoted the wide deployment of cooperative robots in different scenarios like exploration [8], map building [9], search and rescue [10], etc. Meeting the ongoing demands of these applications in terms of memory and computation is a crucial challenge with respect to the limited onboard resources of a robot. Although recent robots are designed with more sophisticated and powerful resources, they are intrinsically resource-constrained because of their size. They cannot meet the explosive growth of applications' requirements in terms of processing and memory. Additionally, computation-intensive tasks could lead to excessive and inappropriate energy consumption. Robots are constrained by their processing and storage capabilities and battery lifetime, which can affect the performance of the application.

Computation offloading [11] can alleviate the challenges of robots in terms of processing to provide an efficient and reliable service (see Fig. 1). It consists of migrating heavy computation applications from robots to remote servers via the Internet. Computation offloading has become a commonly used technique to improve robot processing capabilities. However, computation offloading is worthwhile only if some objectives are achieved. In this context, several research works have evolved towards suggesting new approaches to decide (1) what is the main objective behind the offloading, (2) what is the best location for computation according to the available resources, and (3) what is the best optimization technique to achieve the adequate tasks and computing resources assignment. The offloading decision adapts the computation offloading to different run-time conditions. It is made by analyzing network conditions like fluctuating bandwidth, the amount of data exchanged between computing units, and the state of remote resources like availability, server loads, processing speed, etc. This new vision of computation offloading will:

- Satisfy the QoS of robotic applications. Mobile robots are deployed in applications that are very sensitive to latency, like military or emergency applications. For instance, a robot may need to know obstacles' locations before it collides. If its processor is too slow it can fall into a catastrophic situation.
- Solve the trade-off between the battery lifetime of the robot and the application completion. One of the first objectives behind computation offloading is to save the energy consumption of the robot. Unfortunately, we have to deal with high latency issues in time-critical applications. Using Offloading decisions, we can satisfy both requirements: efficient energy consumption and QoS of robotic applications in terms of latency.
- Enable efficient cooperation between robots. As robots move from one location to another, several parameters may change, such as the network bandwidth, the available external computing resources, etc. The offloading decision can manage with the dynamicity of the environment to guarantee the QoS of robotic applications.

Recent computation offloading solutions for robots depend on several factors [12]. First, the objective of the offloading process is the primary parameter to be considered. Originally, computation offloading was essentially invoked to minimize energy consumption for mobile robots. However, more recently, researchers have considered other conflicting metrics, such as real-time guarantees [13] and the QoS of robotic applications [14]. Second, the location of remote computing resources has a great impact. Cloud-based models were proposed as the first solution for computation offloading of robotic intensive tasks [15]. For example, DAvinCi [16] was the starting computation offloading solution designed based on cloud robotics. Today, several other models were proposed to bring powerful computing resources closer to mobile robots, including edge [17] and fog computing [18]. Finally, researchers have considered different optimization techniques for computation offloading decisions, including heuristics [19], meta-heuristics [20], machine learning [21], and game theory [22].

In this paper, we focus on the offloading decision as it has great potential to improve the computation offloading process. Given the mobility requirement of many robotic applications such as disaster management [23], air and ground transportation [24], healthcare assistance [25], robots are free to move from one location to another. This free mobility raises a new challenge for robots. When moving from one environment to another, robots may lose connection with other network units. Thus, because

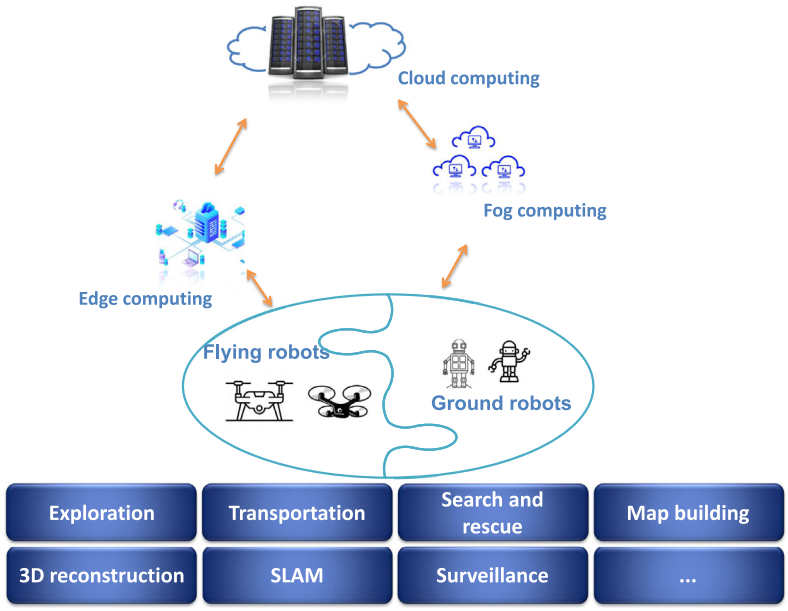

Fig. 1. Computation offloading for flying and ground robots.

of the mobility of the robot, the communication network will change bringing new challenges to QoS guarantees. On account of this, dynamic computation offloading of robotic applications should consider the change in network conditions to improve the quality of robotic applications. For that reason, we specifically discuss the factors affecting the computation offloading for robots, which we turn into three categories: (1) optimization metrics (i.e., objectives), (2) computing models (i.e., cloud/fog/edge), and (3) decision approaches (i.e., algorithms). Playing on these factors will adapt the robotic application to unexpected changes in network conditions.

The contributions of this paper are as follows:

- We first present the computation offloading steps. We mainly focus on offloading decisions for mobile robots; and discuss the factors influencing the computation offloading. We define a taxonomy to identify the factors that influence the offloading decision in the context of robots.
- We provide an overview of the commonly used optimization metrics in computation offloading for robots. We returned them to three primary metrics: energy consumption, completion time, and cost.
- We discuss different computing models used in computation offloading, namely, cloud/fog/edge, and examine their advantages and drawbacks. Moreover, we present the optimization algorithms used in the context of the offloading decision for mobile robots.
- We survey existing research on the computation offloading for robots, mainly architecture solutions and offloading decisions. Moreover, we compare these works according to optimization metrics, computing models, decision approaches, evaluation metrics, etc.
- Finally, we discuss open research challenges and highlight future trends in the offloading decision for robots.

The remainder of this paper is organized as follows. In Section 2, we present the research methodology. Then, we present the related survey in Section 3. In Section 4, we explain the computation offloading concepts. We discuss our proposed taxonomy on offloading decision methodology in Section 5. We review the existing computation offloading architecture and offloading decision in Section 6. Finally, we highlight future directions in Section 7 and summarize lessons learned in Section 8.

2. Search methodology

To search the required papers discussed in this survey, we used the "Google Scholar" search engine. We exploited papers published from commonly known libraries including IEEE Xplore, Springer, ScienceDirect, Elsevier, etc. For finding surveys for the "Related work" section, we used the following search strings:

- allintitle: Computation offloading survey
- allintitle: Computation offloading review
- allintitle: Computation offloading overview
- allintitle: Computation offloading taxonomy

In the "Related work" section, we eliminated papers that present a framework, an architecture, or an offloading solution. Besides, we did not consider papers that discuss specific cyber–physical system like smartphones. Papers, discussed in sections "Architectures and frameworks" and "Offloading decisions", are selected based on some criteria. For example, thesis, patents, and letters were removed. Besides, papers, that present a static offloading solution, were not discussed in the "Offloading decisions" section. The search keywords used to find papers are:

- allintitle: Computation offloading architecture
- allintitle: Computation offloading framework
- allintitle: Computation offloading robot
- allintitle: Computation offloading drone
- allintitle: Computation offloading UAV

The paper selection is a three-step process:

Phase 1: We automatically search papers related to the subject using the keywords previously described.

Phase 2: We select the papers according to some criteria. Our paper is limited to papers (limited to journal and conference papers) published in the last five years. The paper should also discuss work related to mobile robots (whether ground or flying). Besides, papers that are not written in English are omitted. Even low-quality conferences are not considered for discussion in this paper.

Phase 3: The abstracts, introductions, and conclusions of the selected articles are carefully analyzed and verified for adequacy and relevance of the articles to the subject. Based on this selection, each paper is excluded or included in the paper.

3. Related survey

In recent years, several research works have been concerned with offloading of computations from different perspectives (Table 1). Bhattacharya et al. in [12] surveyed the existing techniques used to fit the variations in parameters that influence the computation offloading, including the network conditions and the application to be offloaded. The authors in [26] presented a comparative study of computation offloading frameworks. In [27], the authors surveyed the artificial intelligence approaches used in computation offloading; and highlighted the advantages and limitations of each approach. They deeply analyzed the algorithms used in the case of edge vehicular computing. Shakarami et al. in [28] presented a survey of stochastic-based computation offloading approaches that have been applied for different system models. They classified the criteria of stochastic-based offloading into a taxonomy. Then, the Markov process, Markov chain, and Hidden Markov classes were compared based on performance metrics, use cases, assessment tools, and advantages and disadvantages of each approach.

Several works have surveyed computation offloading in mobile edge computing. For instance, the authors in [29] explained the concept of Mobile Edge Computing (MEC) and the challenges of computation offloading in these models. Shakarami et al. [30] presented the machine learning techniques applied in computation offloading in mobile edge computing models. They mentioned the advantages and drawbacks of each technique and discussed the challenges they faced. Lin et al. [31] explained the concept of mobile edge and the challenges of an edge computing architecture for computing offloading, particularly task partitioning and resource allocation. In [32], the authors surveyed works related to edge computing and the benefits of collaboration between edge and cloud computing to improve the offloading process of computation. The authors in [33] reviewed game-theoretic computation offloading solutions in mobile edge computing. They proposed a classification of these solutions; and compared them to some metrics, including case studies, evaluation tools, etc. In [34], Dubey and Meena reviewed computation offloading approaches in MEC systems. They compared solutions related to this subject and presented challenges faced by computation offloading in MEC systems. In [35], the authors reviewed resource management approaches in a fog computing environment. They classified them into six categories: task offloading, application placement, resource scheduling, load balancing, resource allocation, and resource provisioning techniques. They have discussed the advantages and weaknesses of each category and confirmed that task offloading is the most used resource management in a fog computing environment.

Mobile cloud computing was reviewed by several researchers. In [36], Wu discussed multi-objective offloading decisions to save both energy consumption and response time for mobile cloud computing, considering multiple challenges, including network connectivity and heterogeneity of computing resources. In [37], Kumar et al. discussed the challenges of computation offloading in mobile cloud computing. Akherfi et al. [38] explained the mobile cloud computing concept. They also reviewed existing computation offloading frameworks for mobile devices and discussed the challenges that face these frameworks. In [39], the authors classified works related to computation offloading in MCC into four optimization considerations: objectives, the number of end devices to be served, partition granularity, and the type of offloading. In [40], the authors presented the features of computation offloading in mobile cloud computing (MCC). They surveyed solutions in the context of computation offloading in MCC, focusing on algorithms used in the offloading decision.

A mobile robot, contrarily to a mobile device, can autonomously move from one location to another without any external assistance [41]. Accordingly, computation offloading conditions will change, such as the network bandwidth and available computing resources. Subsequently, we have to cope with three offloading challenges: where to offload, how to offload, and what metrics to be controlled. In this paper, we will focus on dynamic computation offloading for ground and flying robots. Especially, we will discuss the decisions influencing the computation offloading for mobile robots.

4. Computation offloading concepts

Nowadays, applications requiring massive computations, like deep learning tasks, can be remotely accomplished in more powerful computing infrastructure. Mobile robots can overcome restricted resources by migrating computation-intensive tasks to remote servers. Consequently, the main operation of robots would be limited to capturing and offloading data to remote computing infrastructure for rapid computation and analysis. Robots would be updated and actuate accordingly when the remote computation is accomplished.

Offloading robotic applications for remote computation may face challenging issues due to the mobility requirement. Unfortunately, robot mobility can significantly degrade the quality of the computation offloading process. This may cause a variation in network connectivity, increasing the network delay or losing connection with remote computing servers. Therefore, it is crucial to consider these variations to make the offloading process worthwhile. For that, robots need to decide on the main objective behind offloading, whether the network conditions allow offloading or not, and where to offload.

Computation offloading is made of two main steps: (*i.*) application (or program) partitioning and (*ii.*) offloading decision [38]. Application partition consists of identifying the offloading parts of the program or application. Actually, not all tasks can be offloaded. For instance, tasks that require collecting data using a sensor or a camera should be executed locally on the robot; and are not considered for offloading. Unlike voice recognition, image processing, and artificial intelligence tasks, for example, are good candidates for offloading; since they require more computing resources [42].

Application partitioning consists of decomposing the application to offload to multiple small components [32]. The granularity level of this decomposition is determined by some privacy concerns or performance overhead, including CPU consumption, memory utilization, etc. [43]. Then, on the basis of the defined granularity, we model the application to offload. A typical approach for application partitioning for robotics consists of presenting the program as a graph, where vertices represent the program divided into tasks (or functions), and the edges represent the links between them (the data exchanged between functions) [44,45]. Other works rely on queues and Markov theory [46] to present the flow of tasks. Recently, the game-theoretic approach [47] is being used to present the application like a game, in which an equilibrium should be achieved to satisfy all players.

The offloading decision aims at finding an optimal assignment of tasks among the available computing resources. This operation is influenced by mainly two parameters: the *mobility of the robot* and the *network bandwidth*, and it includes several decisions, which we will detail in the next section.

5. Offloading decision methodology

Computation offloading is a collaboration between robots and remote computational resources (edge/fog/cloud) to improve their performance. A critical step in dynamic computation offloading is how to make the offloading decision. Deciding which portion of the application to offload and where to be executed (remotely or locally) is a crucial problem to tackle, especially with the variable

Table 1
A summary of related surveys and their contributions.

Work	Year	Scope	Contribution
[12]	2016	Mobile cloud systems	Techniques used to face variations in parameters influencing the computation offloading
[26]	2016	Mobile cloud computing	Comparative study of computation offloading frameworks
[29]	2017	Mobile edge computing	The challenges of computation offloading in mobile edge computing
[36]	2018	Mobile cloud computing	Multi-objective offloading decision for optimizing energy and completion time
[37]	2018	Mobile cloud computing	The challenges of computation offloading in mobile cloud computing
[38]	2018	Mobile cloud computing	Comparative review of existing computation offloading frameworks
[39]	2019	Mobile cloud computing	Classification of computation offloading works
[31]	2019	Mobile edge computing	The challenges of computation offloading in mobile edge computing
[32]	2019	Cooperation between edge and cloud computing	The improvements made by the collaboration between cloud and edge computing on computation offloading
[27]	2020	Mobile edge computing	The limitations and advantages of artificial intelligence approaches used in computation offloading
[30]	2020	Mobile edge computing	Comparative study of machine learning techniques used in computation offloading
[33]	2020	Mobile edge computing	A review of computation offloading solutions based on game theory for mobile edge computing
[28]	2020	Mobile computing	The computation offloading for mobile devices from a stochastic perspective
[34]	2020	Mobile edge computing	A review of computation offloading approaches in mobile edge computing
[35]	2020	Fog computing	A review on resource management approaches in fog computing environment.
[40]	2021	Mobile cloud computing	Review of computation offloading solutions in MCC, focusing on algorithms used in the offloading decision
Our work	–	Offloading decision for robots	The factors affecting the offloading decision

network conditions. Offloading is heavily influenced by the network condition and robot mobility, which impact communication between robots and remote resources [48].

Computation offloading is made of two main steps: (*i.*) application (or program) partitioning and (*ii.*) offloading decision [38]. The application partition consists of identifying the offloading parts of the program or application. Not all tasks can be offloaded. For instance, tasks that require collecting data using a sensor or a camera should be executed locally on the robot; and are not considered for offloading. Unlike voice recognition, image processing, and artificial intelligence tasks, for example, are good candidates for offloading; since they require more computing resources [42].

Application partitioning consists of decomposing the application to offload to multiple small components [32]. The granularity level of this decomposition is determined by some privacy concerns or performance overhead, including CPU consumption, memory utilization, etc. [43]. Then, based on the defined granularity, we model the application to offload. A typical approach for application partitioning for robotics consists of presenting the program as a graph, where vertices represent the program divided into tasks (or functions), and the edges represent the links between them (the data exchanged between functions) [44,45]. Other works rely on queues and Markov theory [46] to present the flow of tasks. Recently, the game-theoretic approach [47] is being used to present the application like a game, in which an equilibrium should be achieved to satisfy all the players.

The offloading decision aims at finding an optimal assignment of tasks among the available computing resources. This operation is influenced by mainly two parameters: the *mobility of the robot* and the *network bandwidth*, and it includes several decisions, which we will detail in the next section.

Offloading decision is about determining under which circumstances offloading is worthy. However, several factors affect the offloading task, which we classify into the following three factors: (1) the optimization metrics or objectives to be optimized, (2) the optimization approaches, and (3) the computing models. Fig. 2, summarizes these offloading decisive factors.

5.1. Optimization metrics

Over the past years, several research works have focused on improving the Quality of Service (QoS) of mobile devices, including energy consumption [49], completion time [50], cost [51], and delay [52]. The choice of the metric to optimize depends on the application requirements.

The optimization metrics that are commonly utilized with robotic scenarios can be summarized as follows:

- **Energy consumption** is among the primary target optimization metrics. It depends on both computation and communication [53]. In the case of a computation offloading scenario, communication will be the major energy dissipation task due to the transmission of heavy data (e.g., a video stream) from the robot to remote computation units. In the case of local processing, onboard computation will be the main source of energy consumption. Therefore, it is important to make the best trade-off between communication and computation to maximize the overall benefit.
- **Completion time** is another optimization metric that is frequently addressed in the offloading decision. From a computation offloading perspective, the completion time is the duration between sending data to the cloud and receiving the result. For computation-intensive applications, it is crucial to reduce execution time and improve responsiveness. When the execution time increases for computation-intensive tasks, the best strategy would be to offload these

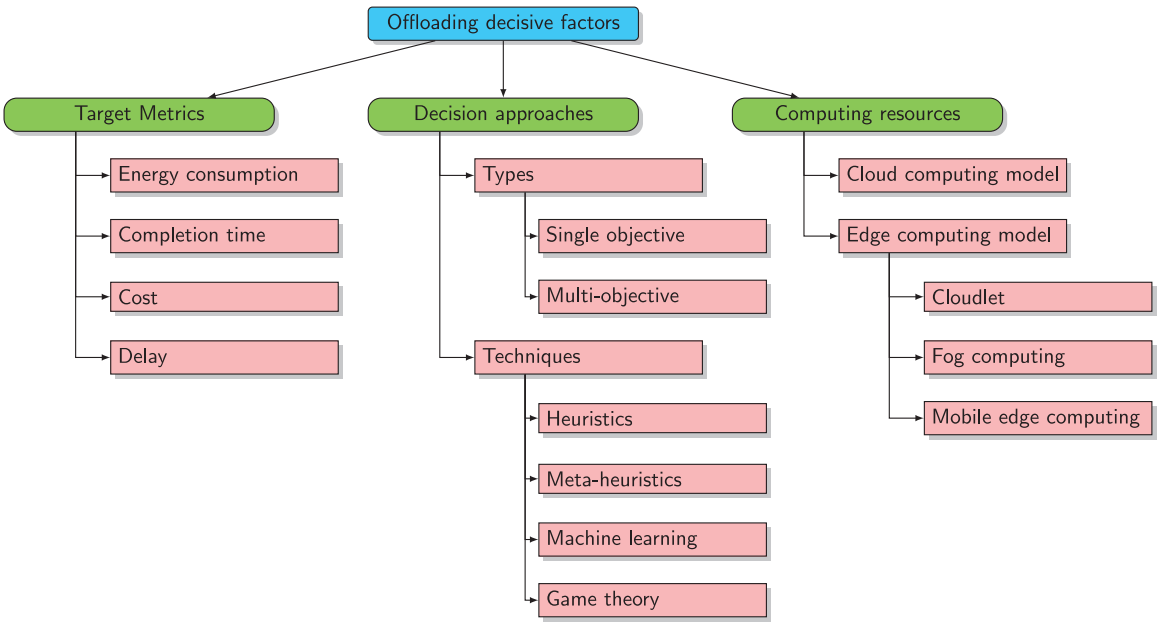

Fig. 2. The main parameters involved in computation offloading decisions.

tasks to the cloud to take advantage of its high-speed processing capabilities. However, communication delays also play an adversary role in achieving real-time performance [54].

- **Delay**, as a metric, is quite challenging, since it cannot be easily formulated in an expression. The computation offloading problem depends on the network communication and task scheduling to different computing destinations. Nevertheless, we distinguish two types of delay: (i.) propagation delay which includes sending the request, queuing, and receiving the response, and (ii.) computation delay [55].
- **Cost** cannot be limited to a single definition. Actually, there are multiple definitions of the metric cost. Each research paper defines the cost according to its requirements. For instance, Afrin et al. in [51] defined the cost as the monetary cost required to execute the workflow. In [56], the authors define the cost as the computational cost of the execution of the tasks. In [57], the authors specify the cost of a task as energy consumption, execution latency, and the possibility of a loss of a task.

To summarize, the offloading decision for robots is concerned with optimizing a single or a combination of multiple optimization metrics from the metrics described above.

5.2. Decision approaches

The offloading decision for mobile and autonomous robots can be considered as an optimization of the available resources to offer the best attainable QoS for robots. If we see the problem from a performance perspective, the optimization is about finding certain values of some of the metrics (described in the previous section), which results in achieving the objectives of a given system concerning some constraints.

The offloading problem is mathematically modeled according to the requirements of the application. It is done using a mathematical formulation of the optimization metrics. Based on the number of metrics involved, we distinguish two optimization models:

- The single objective optimization: mainly focuses on optimizing a single objective function (optimization metric). It consists of defining a single objective function of the problem (Eq. (1)).

$$\arg\min_{x \in X} f(\vec{x}) \tag{1}$$

- Multi-objective optimization: in the context of Multi-Objective Optimization (MOO), the solution is defined as a set of optimal solutions called non-dominated/Pareto-optimal solutions. Classically, the set of Pareto optimal solutions can be found using scalarization techniques [58], which consists of converting the MOO problem to a single objective optimization problem. Multi-Objective Evolutionary Algorithms (MOEAs) [59] were also used to find an approximation of Pareto optimal solutions in MOOP for two or more conflicting objective functions.
  Generally, an MOO is mathematically defined as follows [60]:

$$\text{minimize} \quad F(\vec{x}) = f_1(\vec{x}), f_2(\vec{x}), \ldots, f_o(\vec{x}) \tag{2}$$

$$\text{subject to:} \quad g_l(\vec{x}) >= 0, \quad l = \{1, 2, \ldots, d\} \tag{3}$$

$$h_l(\vec{x}) = 0, \quad l = \{1, 2, \ldots, e\} \tag{4}$$

$$Y_l <= x_l <= Z_l, \quad l = \{1, 2, \ldots, f\} \tag{5}$$

  In fact, a MOO is a minimization problem, in which solutions should satisfy more than one objective function (criterion). It is about finding the set of feasible solution $\vec{x}$ of o objective functions (Eq. (2)), that comply with d inequality constraints $g_l(\vec{x})$ (Eq. (3)), e equality constraints $h_l(\vec{x})$ (Eq. (4)), and f boundaries of the variable x_l (Eq. (5)).

After the mathematical presentation of the offloading problem, it is crucial to choose an efficient decision technique/algorithm to ensure that we obtain the required solution. Decision approaches have a significant influence on the offloading decision. By having all the parameters related to the computation offloading problem, decision approaches will attribute tasks to the available tasks according to the problem description; thus making it possible to achieve the desired objective of the computation offloading. There are no single approaches to fit all the offloading problems. The decision approaches used in offloading computation for mobile robots are:

- **Heuristics**: are a set of rules designed to solve a specific problem [61]. They are usually adapted to a specific problem and do not work for other problems. However, they are often too greedy. They may get locked in a local optimum when searching for a global optimum.
- **Meta-heuristics**: are methods that are designed to solve a wide range of problems [62]. They are applicable for problems containing one global optimum and multiple local optima. In general, they are not greedy and have mechanisms to avoid getting trapped in a local optimum. Making decisions with a metaheuristic technique requires some adjusting to its intrinsic parameters to adapt it to the problem.
- **Machine learning**: consists of programming systems to optimize a performance criterion based on observations or data [63]. It analyzes data, learns from it, and decides what it has learned. In machine learning, we define a model, which can be a mathematical representation of a real-world process. This model is generated using a machine learning algorithm by providing a training data set to learn from it. Deep reinforcement learning [64] is the most used machine learning technique in offloading decisions for mobile robots. Deep learning is a subset of machine learning. A deep learning model is structured as layered algorithms called Artificial Neural Network (ANN), inspired by the biological human brain neural network [65].
- **Game theory**: is a mathematical approach used to study conflict and collaboration between two or more rational decision-makers [66] (also called players). It uses mathematical models to reach whether a Nash Equilibrium or a global optimum [67]. In fact, the main concepts of game theory are inspired by ordinary games. They involve multiple players, a series of rules, and strategies. Each player follows a strategy that presents a benefit for itself and a loss for the other players. Each action is associated with a payoff, which defines whether the action was beneficial or detrimental for each player. Each player acts towards maximizing its payoff. The game evolves to a stationary state, called the Nash Equilibrium. A Nash Equilibrium is a group of strategies where no player can profit by modifying his strategy while the other players' strategies are kept unchanged.

Heuristics and meta-heuristics are useful in looking for an optimal solution. But heuristics are problem-dependent solutions. Game theory offers an optimal solution in situations where players are interdependent. It is recommended rather than meta-heuristics and heuristics in optimizing large-scale search space [68]. Machine learning analyses input data to predict the output. It allows learning how to optimize and improve the offloading decision.

Game theory and machine learning use almost closer concepts like actions, rewards, etc. The main difference between both is that the decision in machine learning is taken by a single agent. However, the decision in game theory is a multi-agent process. Furthermore, the state space in machine learning is infinite, while game theory uses a finite one.

Decision approaches went further from using a single technique. Recently, several research works proposed a combination of two optimization approaches to deal with more complicated problems. For instance, authors in [69] surveyed the works related to the combination of meta-heuristics and machine learning in the healthcare domain. In [70], Tian et al. surveyed meta-heuristic algorithms used for data training in deep learning.

5.3. Computing models

Cloud computing. The idea of computation offloading first appeared with cloud computing [11]. In particular, cloud robotics was a great opportunity for robots to improve their performance and enable new functionalities by discharging computation-intensive tasks from robots to the cloud [71].

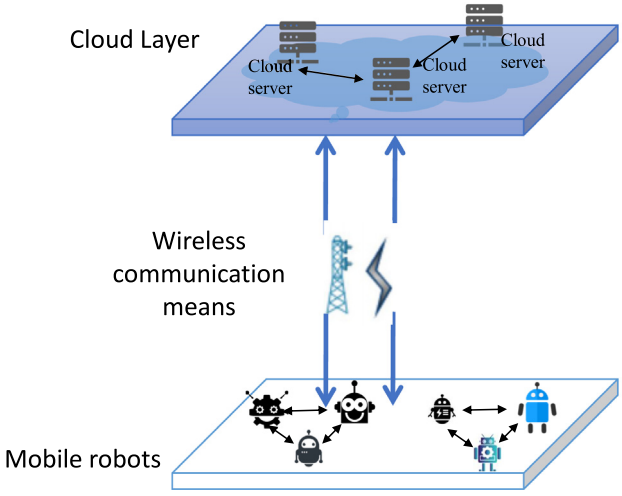

Fig. 3. The structure of the cloud computing model.

The robotic cloud gives access to any robot or automation system to access a cloud infrastructure for data storage or for computation [72]. To get access to cloud services, a user must conclude a contract, known as the Service Level Agreement (SLA), with the cloud service provider [73]. Furthermore, a connection must be created between mobile robots and remote cloud servers to enable data exchange. By doing so, the robot can utilize cloud services accessed by virtualization and offload tasks for on-demand computation.

The cloud computing model is an infrastructure where data storage and computation occur outside the robot, as explained in Fig. 3. It comprises three main components: robots, wireless communication means, and the cloud layer that contains cloud servers [74]. This latter also provides security mechanisms.

Mobile robots can leverage cloud resources [75]. Indeed, cloud computing offers robots increased and powerful computing resources allocated in a pay-as-you-go model. Therefore, robots are not expected to perform computation-intensive tasks on board; instead, they are offloaded to the cloud. This is incredibly considerable for robots to face hardware limitations. Besides, future robots would be able to deal with large volumes of sophisticated data in different types, including captured images in 2D and 3D space, sensory data, etc. Thanks to recent advancements in communications, robots are increasingly becoming capable of capturing and sending collected data to the cloud infrastructure, realizing the concept of the Internet of Things (IoT) [76].

The emergence of robots results in continuous large data transmission in a short time [77]. We live in a hyper-connected world, where communicating and processing the robot's collected data in real-time is a critical issue to be respected. Although cloud robotics was an up-and-coming technology, cloud-based models deploy the processing in cloud data centers. It is not evident to control the network between cloud servers and robots, resulting in high jitter and long delays. Thus, offloading data from robots to the cloud can, unfortunately, fail to comply with the deterministic performance of some robotic applications [78].

Edge computing. Recently, a new computing concept was defined to overcome the limitations of cloud computing models, called *edge computing*. The main idea of this computing paradigm is to bring data storage and computation closer to the robots instead of relying on cloud resources. This computing paradigm

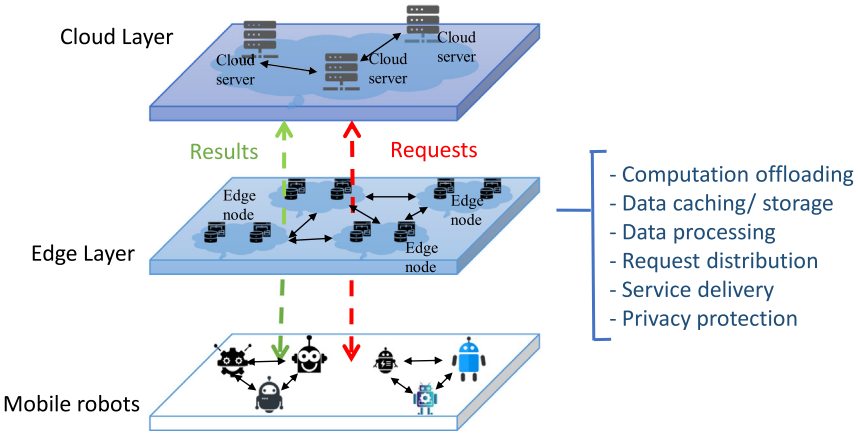

Fig. 4. The structure of the edge computing model.

aims: (i.) to reduce latency by moving the computation of crucial data to the edge of the network, (ii.) to enable data aggregation by collecting readings from different IoT devices, and (iii.) to offer location-dependent and context-aware services [79].

Data exchange in the edge computing model is a bidirectional data stream (see Fig. 4). Each edge node can be at the same time a data producer and a data consumer. Also, the edge node's function is not limited to asking for content and sending a request to the cloud. They can perform data storage, computation, service delivery, etc. However, the edge node should be well designed to meet some requirements, including security and reliability efficiently.

The computational hierarchy in the edge computing model can be organized as follows: the *mobile device*, the *edge layer*, and the *cloud server*, as illustrated in Fig. 4. Three implementations were defined for the edge layer: Fog computing, Cloudlet, mobile edge computing [80].

The Cloudlet is the first implementation of the edge computing model. A Cloudlet [81] is a trusted small-scale data center or a cluster of computers connected to the Internet. They are designed to serve cloud services to mobile devices within their geographical proximity (the Local Wireless Area Network (WLAN)). Later, Cisco systems proposed the concept of Fog computing. According to Bonomi et al. Fog computing can be defined as a "*highly virtualized platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centers, typically, but not exclusively located at the edge of network*" [82]. A fog network is typically composed of traditional networking components like a base station, switches, routers, etc. called fog nodes [83]. Fog nodes, geographically distributed in the proximity of IoT devices, are equipped with storage, computation, and networking capabilities. Consequently, they support the computation offloading of mobile applications and offer cloud-based services to mobile devices. Finally, mobile edge computing, also called Multi-Access Edge Computing, was announced by the European Telecommunications Standards Institute (ETSI) as "*Mobile edge computing provides an IT service environment and cloud computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and close to mobile subscribers.*" [84]. The edge layer in this computing paradigm consists of servers running in base stations of cellular networks. These nodes are provided with storage and computing capabilities to offer cloud computing services inside a RAN.

Both edge computing and cloud computing have similar functionalities as they both leverage computation capabilities for IoT devices. However, the cloud computing model is more scalable than the edge computing model. Scalability in the cloud computing model is concerned with determining how much each cloud instance can carry. While scalability in edge computing is about adding nodes to meet the increasing demand. Besides, with cloud computing, we do not have to invest and manage costly hardware. Instead, we allocate resources in a pay-as-you-go model, which is not the case in edge computing, where we have to create and manage these resources at the edge of the network. In addition, standards were defined to ensure interoperability between different providers and equipment in cloud computing models. Several cloud computing standards organizations are addressing and studying standards in the cloud computing environment.

Although cloud computing is still a good solution for storing and computing data, a new vision is about using edge computing for computation offloading to reduce costs. The fundamental idea is not to replace cloud computing with edge computing, but to separate crucial information from generic ones. Table 2 is a brief comparison between cloud computing and edge computing, that was concluded from these research works [85–87].

6. Computation offloading: state-of-the-art

In this section, we review existing work related to computation offloading for robots. The proposed solutions can generally be classified as architectures and offloading decisions.

6.1. Architectures and frameworks

A fundamental issue to deal with when talking about computation offloading is enabling data exchange between robots and remote computing resources. After data collection, these data should be transmitted to a processing framework in a supported format. This idea has been discussed by researchers, especially with the appearance of cloud robotics.

6.1.1. Ground robots

DAvinCi project [16] is a cloud robotic architecture designed based on the Map-reduce computing model. The critical component of the architecture is the DAvinCi server that acts like a proxy server connecting the two ecosystems, Hadoop distributed File System (HDFS) [88] and Robot Operating Systems (ROS) [89]. It provides a publish/subscribe model for robots to log on to the server and send/update their information. The server sends this data to the backend HDFS. Then, the server executes the data using MapReduce tasks. Finally, these results are communicated to the ROS subscribers. The work was validated by implementing the FastSLAM algorithm and reported significant gains in execution time.

The combination of cloud computing and Service-Oriented Architecture (SOA) [90] technologies enabled the authors in [91] to

Table 2
A brief comparison between edge computing and cloud computing.

	Cloud computing	Edge computing
Latency	High	Low
Internet connectivity	Required	Not required
Number of hops	Multiple hops	Single hop
Standardized protocols	Available	Lack
Service location	Within the internet	In the edge of the network
Scalability	Medium	Low
Hardware	Managed	Unmanaged
Data analysis	Suitable for big data analysis	Suitable for quick data analysis (real-time applications)

carry out robotic applications like object recognition, path planning, and navigation in the cloud using the Map-Reduce [92] computing model. The proposed framework is mainly composed of three components: (1) **Cloud manager** to verify the authorization of clients and receive incoming service requests, (2) **Robotic service handler** to verify the availability of requested applications and execute them, and (3) **Computing cluster** for data processing. The architecture has proved its effectiveness in achieving real-time speech-based navigation

Turnbull in [93] developed a cloud infrastructure to address the problem of forming a multi-robot system. The architecture proposes to use robots designed with minimal hardware, equipped with a microcontroller with WiFi capabilities for both robot-to-robot (R2R) and robot-to-cloud (R2C) communications. The cloud is responsible for controlling the robot's behavior by running the required algorithm. Data collected from a vision acquisition system are sent to the cloud. This latter determines the robot's status and location, controls its behavior, and then sends corresponding commands to the robot.

In [94], L. Bingwei et al. present a Cloud-Enabled Robotics System (CERS) that offers robots the ability to mesh together to construct a local networked system. Robots can offload intensive computation tasks to the external server. This architecture has the advantage of considering security in cloud and robot networks.

Rapyuta [95] designed the ROBOEARTH cloud engine [96] to overcome the limitation of robotic resources by implementing an open-source cloud robotics framework. It uses the Amazon data center to outsource the robots onboard computational intensive tasks by providing a secure, customizable, elastic, and ROS (Robot Operating System) compatible computing environment. This work presents opportunities for a Software as a Service (SaaS) model that may allow customized applications to offload data from robots to the cloud service. This permits to avoidance of a load of executing data on-board the device.

Chaari et al. [97] proposed a distributed cloud robotic architecture for computation offloading based on Apache Kafka [98]. The idea is about designing an offloading middleware acting as the communication engine between robots abstracted by ROS and the cloud servers. This middleware offers seamless, bidirectional, and distributed communication using Apache Kafka messaging brokers. The architecture uses Apache Storm [99] as the cloud computation engine. Through performance evaluation, the architecture presented gains in terms of execution time and power consumption.

Sarker et al. [100] designed a four-layer architecture to offload computation-intensive SLAM tasks: robots, edge gateway layer, fog layer, and cloud. Robots are responsible for collecting information and performing some basic operations. The edge gateway layer collects data from robots and may also execute some computer vision tasks. The fog layer manages the interconnection of edge gateways and the connection between robots and the gateways. The cloud offers administrators or end-users visualization and control tools to monitor robots. The authors demonstrated through a map construction using a controlled car scenario the benefit of using multi-Access edge computing in power consumption.

The authors of [101] proposed an offloading framework for executing the path planning and location application for robots. The framework is mainly composed of robots communicating with an edge server for remote computation. The framework contains also three switching mechanisms, which are: (i.) sensor selection to check the uncertainty in pose estimation, (ii.) estimation offloading in relation to estimating where to execute the localization algorithm based on the computation time, and (iii.) path planning offloading based on the cost of the trajectory. The evaluation of the framework was performed using a single AlphaBot robot with a single edge server. The performance evaluation demonstrated the accuracy in optimizing the completion time compared to local and remote computation.

6.1.2. Flying robots

Dronemap Planner, proposed by Koubaa et al. in [102], is a cloud-based solution for remote management of Unmanned Aerial Vehicles (UAV). It presents a virtualization layer to access, monitor, and control drones and robots using the MAVLink [103] and ROSLink protocols. The architecture was also designed to provide computation offloading services to enhance the performance of UAVs.

Jung et al. [104] proposed an adaptive computation offloading drone system with reliable communication to prolong the operating time of surveillance drones. The system architecture consists of three main modules: response time prediction module, task offloading module, and remote task execution module. The response time prediction module and the task offloading module are run at the drone system while the remote task execution module is run at the ground control system. The response time prediction module takes information related to available resources and wireless conditions factors to decide on computation offloading or running the task locally. In this module, three types of delay including communication delay, input/output access delay, and task execution delay are considered to achieve accurate results. The task offloading module consists of a remote task management service, network service for remote task computing, and data communication. The remote task execution module is responsible for running the tasks on the ground control system and responding to the executed results on the drone system. Additionally, the module updates the ground control system's available resources to the response time prediction module of the drone system. The system is implemented on an actual quadcopter drone and a mobile laptop as a ground control system. The results show that the proposed computation offloading drone system helps to improve the operating time of a drone while reducing the response time.

Qu et al. [105] presented a computation offloading and control framework for drone video analytics. The proposed framework enables trade-offs between processing latency and the performance of drone video analytics. The proposed framework is based on the architecture proposed in [106]. The framework consists of

a decision-making strategy that decides on the proper locations where standalone functions are executed. In this framework, standalone functions are retrieved from the application pipeline, and they can be executed separately in different computing nodes such as edge or cloud. The proposed framework is tested with the GENI infrastructure [107]. The results show that the proposed framework can choose a suitable topology and allocate resources correctly according to latency and bandwidth requirements. Although the proposed framework uses different factors such as latency, network failure, and network bandwidth, it still has limitations. In particular, the energy consumption of the drone is not considered. In addition, the proposed framework was not implemented with actual hardware and drones. Hence, the proposed framework may not be suitable for many applications in practice.

Koubaa et al. [78] proposed system architecture for computation offloading for IoT-based drones. The architecture consists of four main layers such as unmanned system, multi-Access edge, cloud, and end-user layer. By applying the proposed architecture, the users access drones over the Internet and the drones can offload tasks into cloud servers. In case of computation offloading, images captured by a drone are sent to cloud servers for further processing or analysis, and then the results are sent to a user. In the case of executing the tasks on the drones (image processing), the images captured by a drone are processed first at the edge then the processed image is sent to cloud servers that can be accessed by a user for the final results. The result shows that the proposed system can achieve higher throughput (i.e., frames per second) but has large delays for high-quality video. Although the proposed architecture has proved advantages, it still has some limitations. For instance, the entire system cannot work properly if the connection between drones and an edge server is interrupted or disconnected. In addition, the proposed architecture completely relies on cloud computing to perform deep learning approaches. Due to the benefits of edge computing [108,109], the proposed architecture would achieve better results in terms of latency if some parts of the deep learning approaches were carried out at edge servers.

Wang et al. [110] presented an agent-enabled task offloading architecture. The architecture consists of three main layers: user devices, access network, and core network. The user devices are responsible for sensing and transmitting the data wirelessly. The access network layer consists of UAV clustering and MEC servers that communicate with each other via a wireless protocol. The difference between the proposed architecture and others is that user devices can send the data to either MEC servers or UAVs. If the data is sent to UAVs, it will be forwarded to MEC servers which can process the data or sends the data to Cloud servers in a core network layer. In this architecture, UAVs are used to track a user based on the geographical location of the user and provide some micro-services to users. This architecture allows offloading in different places, e.g., at UAVs, MEC, or Cloud servers depending on the situation. This makes the proposed architecture suitable for different applications such as tracking objects.

Luo et al. [111] proposed architecture for offloading user tasks in mobile multi-Access edge computing. The architecture utilizes drones and private blockchain to overcome two existing issues of mobile multi-Access edge computing (MEC), including the limitation of the coverage area of MEC services and the difficulty of developing an audit trail service, which can track MEC service providers processing the user data [112]. In particular, drones are used to dynamically get data sensed by IoT devices and to send the data collected to MEC servers that are part of the private blockchain network. The experimental result shows the flexibility of the task offloading policies. Although the proposed architecture offers many advantages, it still has many disadvantages. For example, drones are used as a hub for receiving and transmitting the collected data to MEC servers. Due to the limited battery capacity, drones cannot fly for a long period of time. Therefore, it is not practical to apply the proposed architecture in many real-life cases.

6.1.3. Comparison

In this section, we present a comparison of various architectures and frameworks for ground and flying robots. In Table 3, we present the advantages and drawbacks of each work. Works are organized by year of publication.

6.2. Offloading decisions

In this section, we provide state-of-the-art offloading decisions for robots.

6.2.1. Ground robots

Rahman et al. [113] proposed a cloud robotic framework for dynamic computation offloading to guarantee the required QoS of smart city applications. In this work, the offloading decision is formulated to optimize energy consumption constrained by a tolerable completion time. Consequently, they proposed an approach based on the genetic algorithm [114] to solve the problem. They evaluated their solution through simulation by measuring the average fitness score and the lowest fitness score achieved by the GA. They proved also that their solution can achieve minimum energy consumption by comparing it with three benchmarks: Exhaustive Search (ES), All-on-Robot (AoR), and the greedy algorithm. Besides, they evaluated the adaptability to the bandwidth change. However, the fluctuation in the bandwidth network cannot be predicted during the offloading decision process. The system is designed with a fixed bandwidth value which is far from real cases. The same authors in [115] formulated the problems of robot mobility and communication in maintenance applications in a smart factory as a joint optimization problem. This problem considers three factors: task offloading decision, the access point (enabling communication with the cloud), and the computed path for robot mobility. For that issue, they adapted the genetic algorithm to optimize the robot's energy consumption constrained by an imposed latency. They proposed a three-layer chromosome encoding to present task offloading, access point selection, and path planning. In their simulation, they proved the effectiveness of the solution in finding the optimal energy consumption. Finally, Rahman et al. [48] suggested another optimization study of the robot's energy consumption in a cloud robotic model. This paper's novelty is considering the robot's mobility, bandwidth variability, and robot to robot offloading in the decision-making process. Consequently, they formulated the problem as a joint optimization, which was resolved by developing a 4-layer genetic algorithm. The performance evaluation of this work focused mainly on measuring the minimum energy consumption. This work is good exploitation of a multi-robot system in real cases since it takes into consideration the robot's mobility, path planning, etc. However, the variability in network bandwidth is not presented with a model to bring it closer to the real world. Instead, it is presented with an access point with different static bandwidth values.

The authors in [116] considered the computation offloading problem as a joint optimization of three metrics: the energy consumption, the latency, and the cost. The optimization problem takes into account the characteristics of networked cloud robots, and the QoS required for robotic workflow including response time. They turned the computation offloading problem into the mixed-integer linear programming [117] optimization model, which they solved by proposing a heuristic algorithm.

Table 3
A comparison between architectures and frameworks for ground and flying robots.

Work	Year	Advantages	Drawbacks
[16]	2010	The collaboration cloud and robots accurate maps shared among multiple robots.	The proposed framework does not address the reliability and communication latency issues of the DAvinCi server between roots and cloud servers.
[91]	2012	The effectiveness of the architecture in executing real-time speech-based navigation.	Much needs to be appreciated in terms of architecture validation while running real-world application scenarios.
[93]	2013	The architecture is designed with minimal hardware.	The effectiveness of the architecture should be validated.
[94]	2014	The architecture considers security in the cloud and robots networks.	More experiments are needed to prove the real-time aspect of the architecture.
[95]	2015	The architecture presents a good opportunity for software as a service model.	
[104]	2017	The adaptability of the proposed solution to parameters like resources availability.	It does not consider changes in network bandwidth.
[102]	2019	The architecture can serve both UAVs and ROS operated robots in real-time.	Security should be investigated to improve the architecture.
[97]	2019	The architecture is designed based on open-source software, reducing therefore the cost of managing different resources.	The architecture introduces latency due to execution of applications in the cloud.
[100]	2019	The architecture reduces the latency of SLAM applications through leveraging computation at three levels: edge, fog, and cloud levels.	The architecture lacks security mechanisms ensuring the protection of transmitted data.
[105]	2019	The framework considers network failure to avoid packet loss.	No performance evaluation was conducted to prove the effectiveness of the framework.
[78]	2020	The architecture can achieve higher throughput.	The architecture introduces large delays for high-quality video. It cannot work properly if the connection between drones and an edge server is interrupted or disconnected.
[101]	2020	The architecture proposed a switching mechanism for optimizing completion time. The architecture showed its accuracy in optimizing the completion time.	More evaluation is needed to prove the stability of the proposed system to network conditions.
[110]	2020	The architecture enables offloading at different levels. It includes a decision making based on environmental perception.	Real experiments are needed to prove the effectiveness of the architecture in reducing delays.
[111]	2021	This work uses blockchain technology making mobile edge clusters trustful.	The architecture surcharges drones with extra tasks consuming energy.

Through simulation, they proved the efficiency of their solution in optimizing the three metrics.

Some works considered task offloading as a Multi-Objective optimization problem. For instance, Afrin et al. in [51] redesigned a new version of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [118] to solve the problem of resource allocation between robots and an edge cloud infrastructure deployed in a smart city application. They defined the problem as a constrained minimization of three optimization metrics: the completion time, the energy consumption, and the monetary cost required for task execution. The proposed solution outperforms other multi-objective algorithms in optimizing the metrics.

Guo et al. [119] present a task offloading framework that allows offloading tasks to the cloud and other robots. They proposed an energy-sensitive GA that considers the energy of all robots to distribute tasks. The proposed algorithm ensures balanced task assignment among robots to avoid battery depletion in some robots, while respecting a tolerable completion time. Through simulation, they demonstrated the efficiency of their approach in balancing the energy consumption of the robots by measuring four performance metrics: decision making time, decision accuracy, network lifetime, and energy consumption standard deviation. This paper considers an important optimization metric which is the standard deviation of the energy consumption. Unfortunately, they did not prove the importance of using this metric in extending the lifetime of the robots while executing an application. Later, Guo et al. [120] proposed optimization of energy consumption and completion time in a cloud robotic model. They presented the robots as a Connected Dominating Set (CDS) [121] to maintain balanced energy consumption among them. Then, they employed the Dijkstra algorithm [122] and the genetic algorithm to find the optimal offloading strategy. In the performance evaluation, they first evaluated the impact of the number and sparsity of robots on the construction of CDS. Then, they evaluated the performance of their approach in achieving optimal scores.

The widespread use of deep learning algorithms reached several application domains, including medicine, big data analysis, etc. Chinchali et al. in [123] mathematically formulated the problem of minimization of completion time in cloud robotic as a Markov Decision Process (MDP) [124]. They used deep reinforcement learning [21] to solve the offloading problem. Through experiments, they proved the efficiency of the deep reinforcement learning method in optimizing the completion time by comparing them with some common heuristics. Experiments proved that the proposed solution outperforms some heuristics in reducing latency. However, this solution does not consider the impact of variation in network conditions on the input streams.

Hong et al. [125] used the game theory approach to the offloading decision problem. They studied the routing problem conjointly with the multi-hop computation offloading in cloud robotics to optimize latency and energy consumption. Consequently, they proposed a distributed QoS-aware solution based on game theory. Compared to other predefined algorithms, the proposed solution was more stable in reaching optimal energy, delay, and path.

Xu et al. [126] proposed an improved game theory algorithm to optimize both energy consumption and completion time. The idea consists of assigning each robot weighting factors to the completion time and energy consumption. They simulated a

cloud robotic model using the OpenStack [127] cloud computing platform, the Arduino platform to create the computing offload scenario, and 50 robots. Experiments showed that the algorithm surpasses some other solutions in terms of convergence and reaching an optimized energy consumption and completion time. They studied also the impact of weight factors on energy consumption and completion time.

Kumaran et al. [128] proposed Hybrid Intelligent Generic Algorithm (HIGA) to separately optimize the total energy consumption and the completion time of a cloud robotic system taking into account the robot's mobility. The system is specifically designed for smart city applications in which robots collect data from sensors and send them to the cloud. Through experiments, they demonstrated that the proposed algorithm outperforms some other predefined solutions in optimizing the energy consumption and completion time of a flow of tasks in smart city applications. They also investigated the impact of bandwidth on the offloading decision and completion time. This result is obvious since increasing the bandwidth will promote offloading tasks to the cloud.

In [129], the authors present a machine learning-based approach for computation offloading and resource provisioning for three-tier architecture systems that have used edge and cloud computing. In particular, the authors use learning automata as the main technique for decision-making of computation offloading, in which workloads are decided to be processed at the edge or cloud servers. Furthermore, deep learning-based techniques are deployed at a cloud server for resource provisioning in which an LSTM model is used to estimate the future workload, and a reinforcement learning technique is used to decide an appropriate scaling. The simulated results show that the proposed approach helps improve CPU utilization, and minimize energy consumption and execution latency.

Authors in [130] modeled the task scheduling problem in a cloud robotic environment as a Markovian decision process to optimize the application execution time. They proposed a deep reinforcement learning approach to solve the problem. More precisely, they assumed that the input data size has a direct impact on the execution time of the robotic application, defined a set of actions for each time step, and trained the outcome using a deep Q network algorithm. The proposed approach was validated using a robot and an AWS cloud server that collaboratively execute a path planning application. They demonstrated the effectiveness of their algorithm and its accuracy compared to other algorithms.

The authors in [131] offer a computation offloading framework for mobile edge computing systems based on the MAPE-K loop approach. The MAPE-K loop is an IBM-defined conceptual control paradigm for self-management systems. Monitoring, analyzing, planning, and executing are the four components of the MAPE-K loop. To solve concerns related to time and resource-intensive, the framework includes an autonomous compute offloading mechanism based on a deep learning hybrid method. Multiple linear regression and deep learning on neural networks (DNN) are used in particular to determine the best sites to execute incoming requests from lower levels. A Hidden Markov Model is also used to choose the best uplink transmission media. The simulated results show that the proposed framework can address the issues related to the prediction of offloading decision, energy consumption, uplink transmission selection with high performance.

Wang et al. [132] developed an offloading solution for a swarm of robots connected to an edge server for remote computation. The main objective of their solution is to optimize the energy consumption and the computation delay. For that, they developed a deep reinforcement learning method that considers the robot's mobility in task scheduling. Through simulation, they demonstrated the effectiveness of their solution in reducing energy consumption and delay. Unfortunately, the system model relies on a single edge server for managing and performing the computation offloading.

In [133], the authors propose a computation offloading and auto-scaling approach for mobile fog computing. The approach utilizes a MAPE control loop to make decisions on the location where application modules should be processed to achieve system efficiency. In particular, a module can be processed locally at a sensor device or offloaded to Fog or cloud servers according to an offloading rate which is analyzed by the Markov Decision Process. When the offloading decision is made to run application modules at Fog or cloud servers, a deep reinforcement learning algorithm is applied to find the most suitable location. Different parameters are acquired, including power consumption, execution cost, latency, and network resource usage, to evaluate the performance of the proposed approach. The results show that the proposed approach performs better than other approaches.

In [134], the authors present a computation offloading approach based on Bayesian learning automata for mobile edge computing. The proposed approach considers context information that impacts computation offloading decisions. The context information is captured via the MAPE control loop. Several metrics including energy consumption, execution cost, network usage, latency, mobile types, module size, and time interval of offloading operations are captured to evaluate their solution.

In [135], the authors present a computation offloading approach for mobile edge computing. The approach aims to address the complexity that is caused by fast changes in context conditions. The MAPE concept, which consists of four components of monitoring, analysis, plan, and execution, is applied in the proposed approach to make offloading decisions. The simulated results show that the proposed approach has a better performance in terms of latency, energy consumption, network usage and execution cost than other approaches including local computation and offloading approaches that does not consider the impact of the operating context.

6.2.2. Flying robots

Jo et al. [136] proposed an offloading solution for a multi-drone system for optimizing both energy consumption and completion time. The proposed scheme uses the cost analysis model to make the decision of task offloading of a crowd surveillance system made up of a cloud server and a ground control center to manage a cloudlet cluster of drones. The proposed scheme is evaluated on the prototype consisting of an actual drone and a mobile laptop that is used as a ground control station. Simulations show that the proposed offloading scheme can help improve energy efficiency compared to the case without offloading. Although the proposed offloading scheme can help to improve the fighting time of the drones, the proposed approach has many limitations. For example, small tasks that are offloaded are not properly mentioned and considered (such as dependent tasks or standalone tasks). The latency for completing the dependent tasks might not be acquired at some moments as it is retrieved from both the latency for executing the tasks and the waiting time that depends on the responses of other tasks. Consequently, an offload decision cannot be achieved at some moments, which can cause some delays.

Swarm drones present a good opportunity for several applications. However, they are still restricted by the available energy and computation resources. In [137], the authors proposed an opportunistic offloading solution to reduce the response time for drones. The solution predicts the response time using artificial neural networks. The decision on whether to execute the task locally or offload it to a cluster of drones takes into

consideration several parameters, including the network channel condition, availability of other neighbor clusters, etc. Through simulations, they demonstrated the accuracy of their solution. This work proposed a good solution to handle changes related to mobility. However, the offloading is managed by the cluster heap. consequently, its absence or disconnection will prevent all the computation offloading processes and cause data loss.

Chemodanov et al. [106] proposed an offloading solution for drone video analytics application. They proposed a framework in which drones are connected to an edge server and a cloud server via RESTful API [138]. In the scheme, a policy-based decision-making mechanism is used to achieve real-time performance at a low cost using a deep learning-based solution. Particularly, the three policies including the real-time control policy, cost/performance preference policy, and the function-centric availability policy are applied. In the real-time control policy, a part of video analytics is run at the edge with low latency while in the cost/performance preference policy, a user can choose the performance level such as low or high performance of video analytics. The proposed scheme with different policies is implemented and evaluated by analyzing metrics such as frame per second, round-trip time, and frame processing cost. The results show different choices of the scheme to achieve high performance when dealing with different requirements.

Yao et al. [139] presented a task-offloading approach for the Internet of Drones to improve the energy efficiency of drones. The authors proposed a joint optimization of the task offloading with the flying speed of the drones in the fog computing model. Accordingly, they formulated the problem as a mixed-integer non-linear programming (MINLP) problem [140], which they solved through a heuristic algorithm. The algorithm is simulated and compared with predefined algorithms. The results show that when battery capacity is not considered, the proposed algorithm can help the drone to complete the assigned tasks in a shorter time. In addition, the algorithm helps to reduce the energy consumption of a drone while maintaining its flying speed at certain levels. Compared with other state-of-the-art offloading approaches, the proposed approach is one of the approaches considering environment-related parameters. This leads the approach to be more suitable for practical applications. However, it would be more practical if the proposed approach considered wind and different heights when measuring drone flying speed and energy consumption.

Kim et al. [141] presented a computation offloading scheme to reduce the energy consumption of AI-based drones, performing deep learning tasks [142] while satisfying time requirements. The offloading scheme tries to find out the best layer where the energy consumption of the drone is minimal by using a cloud server for the execution of some tasks. Two models, including the execution time model and the energy consumption model, have been used to achieve the computing time and energy consumption of a drone when performing a job on a specific layer. The execution time model uses several parameters such as computation time of a job on a drone, data size, network bandwidth, and processing time on servers, while the energy consumption model relies on both energy consumption of processing some parts of a job on a drone and energy consumption of transmitting the rest parts to cloud servers. The proposed scheme was simulated and the results show that the proposed offloading solution can help to minimize the energy consumption of a drone while satisfying the time requirements. Although the proposed approach shows its benefits, it may not be suitable for several applications that do not have powerful drones.

Table 4 summarizes the previously discussed offloading decision solutions for robots.

6.2.3. Comparison

In this section, we provide a comparative study of different offloading decisions for ground and flying robots. Table 5 presents a comparison between the offloading decision works in terms of evaluation tools, performance metrics, advantages, weaknesses, and case studies.

According to the literature review presented in this survey, we present some technical analysis based on the scope of offloading decisions designed for mobile robots. In what follows, we present these results in question/response form.

Q1: What is the most commonly used computing model in offloading decisions for mobile robots?

According to Fig. 5, more than 50% of the research papers use the cloud computing model in their offloading decision solutions. Since resources in the cloud computing model are far from cloud servers, it may introduce high latency. Minimizing both execution time and delay are two factors for finding an adequate offloading solution in the cloud computing model. In addition, mobile edge computing is more exploited than the fog computing model. The edge computing model has the advantage of giving faster results for mobile robots by enabling processing at a local level.

Q2: What evaluation metrics are used for evaluating offloading decisions for mobile robots?

The evaluation metrics applied to offloading decision solutions are depicted in Fig. 6. Some papers focused on QoS specifications like energy consumption. While others concentrated on evaluating algorithm performance including convergence, accuracy, etc. The QoS specifications analytical report reveals that energy consumption and time are the most used metrics. Of course, network usage is a challenging metric in the QoS specification f offloading decision for mobile robots. Algorithm performance is also an important metric for evaluating the proposed solution to ensure convergence.

Q3: What tools are usually utilized for evaluating offloading decisions for mobile robots?

In offloading decisions designed for mobile robots, researchers adopt whether real implementation or simulation. In most cases, simulation is used for evaluation. Most evaluation tools in this context are depicted in Fig. 7. Mainly, three simulators were equally used: Matlab, iFogSim Toolkit, and NetworkSimulator-3. Unfortunately, much research did not mention their evaluation tools.

7. Open research questions

Computation offloading for robots presents several benefits and challenges in real-world deployments. On the one hand, it allows to leverage the computing and storage resources of the cloud to process computation-intensive applications (e.g. deep learning, computer vision), and on the other hand, communication over the Internet poses several issues in terms of delays and energy consumption. In particular, the offloading decision is an essential step in improving the robots' quality of experience and supporting the execution of applications with low latency requirements. In this survey, we first explain the computation offloading and review existing work from an architectural perspective. Then, we discussed three factors contributing to computation offloading: optimization metrics, decision techniques, and the distribution of the computing resources. Furthermore, we conducted a comparative review of recent research efforts on offloading decisions for mobile robots.

Certainly, massive efforts were made to improve the robot's performance through computation offloading. However, several challenges remain open research areas for further investigations. In what follows, we discuss some of these challenges.

[1] In Fig. 6, several metrics were classified as Algorithm performance such as accuracy and effectiveness of the proposed solution.

Table 4
A summary of existing works on offloading decisions for ground and flying robots.

Work	Year	Offloading metrics				Decision approaches		Decision techniques				Computing resources			
		Energy	Time	Cost	Delay	Single	Multi	H	MH	GT	ML	Robot	Edge	Fog	Cloud
[113]	2016	*				*			*			*			*
[136]	2017	*	*				*	*				*	*		
[116]	2018	*		*	*		*	*				*			*
[137]	2018		*			*					*	*			
[115]	2019	*				*			*			*			*
[48]	2019	*				*			*			*			*
[51]	2019	*	*	*			*		*			*	*		*
[119]	2019	*				*			*			*			*
[120]	2019	*	*			*			*			*			*
[123]	2019		*			*					*	*			*
[125]	2019	*			*		*			*		*			*
[126]	2020	*	*				*			*		*			*
[139]	2020	*				*		*				*		*	
[128]	2020	*	*			*		*				*			*
[129]	2020	*	*			*					*	*	*		*
[133]	2021	*			*		*				*	*		*	*
[130]	2021		*			*					*	*			*
[141]	2021	*				*		*				*			*
[132]	2021	*			*	*					*	*	*		
[131]	2021	*			*		*				*	*	*		*
[134]	2021			*		*					*	*	*		*
[135]	2021	*				*		*				*	*		*

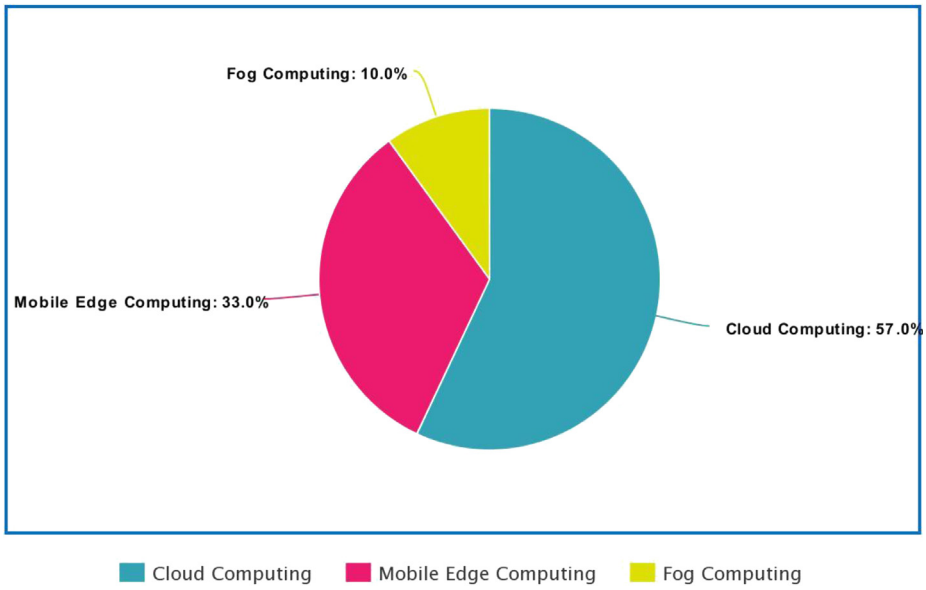

Fig. 5. The percentage of computing models used in the decision to load mobile robots.

7.1. Security

Security [143] is an important dimension of performance in all networked robotic scenarios [144]. Communication and transferring data between robots and remote computing resources can result in some security threats, namely [145]: alteration, denial of service, and information disclosure [146,147]. By way of example, an attacker executes illegal tasks that may prohibit other tasks from accessing intermediate data, which may result in damages to the execution of the application. Furthermore, a malicious task that modifies the intermediate data exchanged in the network may affect the task receiving that data or the computing resources (whether a robot or a server) running that task [148]. Thus, reducing the number of security threads is an important issue that should be considered to provide a more secure environment to run robotic applications [149]. Unfortunately, security was not considered as an optimization metric in the works mentioned above. In the context of the deep learning applications, some techniques known as privacy-preservation deep learning methods based on Homomorphic encryption [148,150,151] are emerging and are based on encrypting the data before sending it to the cloud for processing in both the training and prediction phases. These techniques can also be incorporated into cloud or

Table 5
A comparison between existing works on offloading decision for ground and flying robots.

Work/ year	Performance metrics	Evaluation tools	Case study	Advantages	Weakness
[113] 2016	Energy consumption Adaptability to changes in bandwidth	Simulation Matlab	Smart city	Cost model take into consideration mobility	No consideration for bandwidth variability
[136] 2017	Energy consumption Response time	Simulation	Surveillance	Improve the fighting time of the drones	Causes delays
[116] 2018	Operating cost Execution time	Simulation	General applications	The effectiveness of the proposed algorithm in optimizing three optimization metrics: energy, execution time, and cost Consider task parallelism	Ignore the limitation of network resources
[137] 2018	Response time Error prediction	Simulation NetworkSimulator-3	General Applications	Enable offloading to other drones	No consideration for energy consumption
[115] 2019	Energy consumption Robot performance	Simulation	Smart factory	Path planning and access point selection are considered in the offloading decision	Theoretical results under these conditions
[48] 2019	Performance of GA Robot performance	Simulation	Smart factory	Considering the robot's mobility, bandwidth variability, and robot to robot offloading in the decision-making process	Results are too theoretical
[51]2019	Average energy, makespan and cost	Simulation Matlab	General applications	The proposed solution optimizes three metrics The effectiveness of the modified version of NSGA-II	The implementation of the model is too expensive in terms of resources
[119] 2019	Variation in standard deviation	Simulation	General applications	Consider a balanced energy consumption between robots	No performance was conducted to prove the efficiency of the proposed solution in extending the robot's lifetime.
[120] 2019	Effect of robot sparsity Network life analysis	Simulation	General applications	The solution prolongs the robotic network lifetime	Execution-intensive solution
[123] 2019	Latency	Experiment	General applications	The proposed solution outperforms some heuristics in reducing latency	It does not consider the impact of network conditions on input stream.
[125] 2019	Performance gain Profit ratio	Simulation	General applications	The convergence and efficiency of the proposed offloading solution	Centralized system model
[126] 2020	Convergence Cost	Implementation	General applications	Powerful mathematical presentation of the problem	No consideration of task scheduling on computing resources
[139] 2020	Energy consumption Completion time	Simulation	General applications	Effectiveness of the online-based offloading algorithm in optimizing both energy consumption and completion time	No consideration of wind and different heights when measuring the flying speed of the drone and energy consumption
[128] 2020	Energy consumption Time Adaptability to changes in bandwidth	Simulation	Smart city	Considering the robot's mobility	Obvious results
[129] 2020	Execution time Energy consumption CPU utilization	Simulation iFogSim Toolkit	General applications	They propose learning-based solution for optimizing both energy and time.	The proposed solution charges the offloading decider edge with additional processing.
[133] 2021	Energy consumption Total execution cost Total network usage Application delay	Simulation iFogSim	Smart surveillance	Find better compromise between energy consumption and delay	High complexity and low convergence algorithm
[130] 2021	Effectiveness Accuracy	Implementation	Path planning applications	A learning solution for robots to reduce latency based on input data size	Experiments were conducted with minimum hardware.
[141] 2021	Execution time Energy consumption Deadline meet ratio	Implementation	Deep learning	Meet real-time requirement	Requires powerful drones.
[132] 2021	Average performance	Simulation (python)	General applications	The solution considers the robot's mobility	Use of a single and central edge server
[131] 2021	Delay Accuracy Energy consumption	Simulation (python)	General applications	The proposed algorithm can find an optimal offloading decision solution with a good prediction of the latency, up-link selection, and energy consumption.	The proposed solution is expensive in terms of time and resources.

(*continued on next page*)

Table 5 (*continued*).

Work/ year	Performance metrics	Evaluation tools	Case study	Advantages	Weakness
[134] 2021	Energy consumption Total cost Network usage Delay Jain index	Simulation Toolkit iFogSim	Virtual reality Game	The use of context-aware information to improve the network performance. A multi-used solution.	Low convergence and high complexity algorithm.
[135] 2021	Energy consumption Total execution cost Total network usage Application delay	Simulation Toolkit iFogSim	Virtual reality game	The use of context information in the offloading decision.	The results regarding improving the network performance remain theoretical because simulation is far from presenting real network conditions.

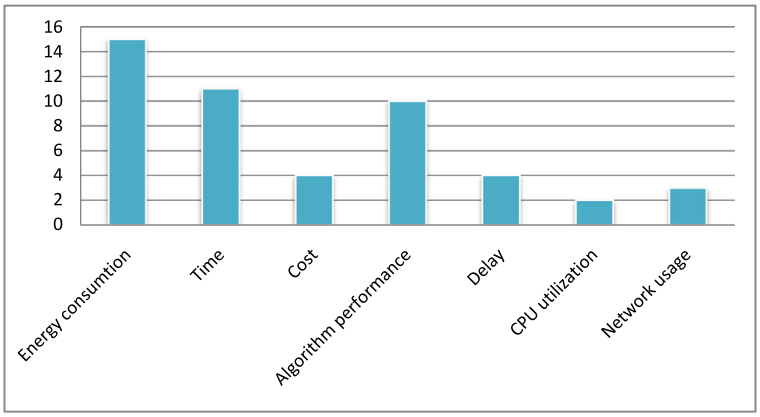

Fig. 6. The evaluation metrics used for evaluating offloading decision for mobile robots [1].

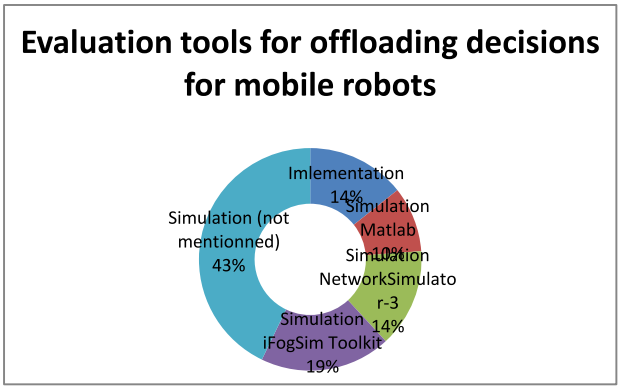

Fig. 7. The percentage of evaluation tools used in offloading decision.

edge robotics for securely offloading deep learning applications to the cloud without having to reveal the data.

7.2. Context awareness

The offloading solution would be more efficient if it can react to changes in the network condition, resources availability, robot's location, etc. Unfortunately, despite all the efforts made in improving the offloading process, fluctuating network conditions, for example, is still a critical issue that was not investigated in most of the offloading decision solutions. Some works, like [48,78], considered variability in bandwidth. However, most of the existing works assume that the wireless channels remain static during the task execution. This assumption is right only if the channel coherence time is larger than the latency requirement, which is not always the case.

Context-awareness is a beneficial feature that should be investigated; since we are talking about dynamic computation offloading. It should be considered during all the computation offloading steps. For instance, the program partitioning step needs to know about the requirement in terms of the resources of the application to offload. Then, the offloading decision should be every time informed of the context information, including remote servers availability, network quality, etc.

7.3. Connectivity loss

The worst thing that can happen during offloading is losing connection with remote computing resources. Imagine a robot is busy executing some tasks and offloading some others for computation. Unfortunately, it loses connection with remote resources. Accordingly, the robot will take charge of the execution of all the tasks, increasing, therefore, the application delay and information loss.

Dew robotics [152] is a new model where a few services are available for mobile robots even in the absence of connectivity (see Fig. 8). In the Dew framework, a cache is created of the lastly retrieved information. This cache can be used to store information in case remote resources are unreachable. With Dew robotic model, we can exploit robot-to-robot connection and offload some urgent tasks to other available robots.

7.4. Fault tolerance

One of the most critical challenges in the computation offloading for mobile robots is the dependency on the network conditions. This may cause a malfunction of the offloading process, which would be either because of poor connection between robots and remote resources or a problem related to the remote server. Since mobile robots are supposed to run in critical applications, like the military, it is crucial to avoid these failures, and guarantee a successful execution of the application.

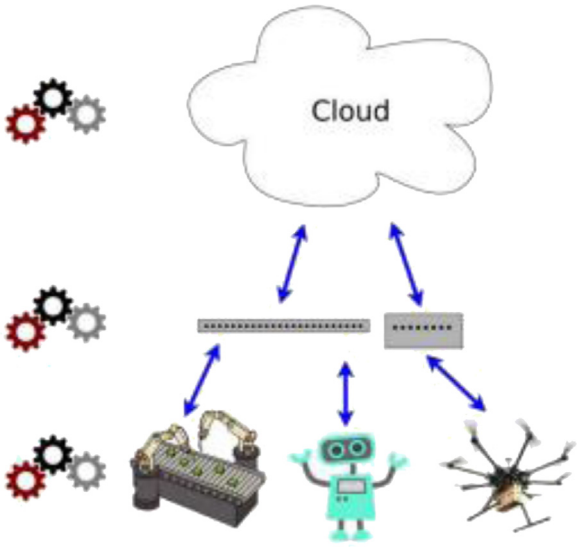

Fig. 8. Dew robotics computing model [153].

When a connection problem happens, the mobile robot should be informed by other alternatives. For instance, it may choose to execute the application locally. It may also offload it to other possible computing resources like an edge server available in proximity.

All the errors that can happen during the offloading, like losing connection to the server and servers switching, come with extra energy consumption and execution time. An effective offloading solution should consider the cost of these errors and the recovery operation as well. Unfortunately, all the available frameworks focus only on the offloading solution without either a recovery process or a failure handling.

8. Lessons learned

In this paper, we considered computation offloading for flying and ground robots. Our work is mainly focusing on the decision influencing the offloading process. We provided also a review of existing offloading solutions for mobile robots. The lessons learned by this survey are:

- The cooperation between ground and flying robots has enhanced the execution of many applications. Through dynamic computation offloading, this collaboration has been reinforced with more computing resources.
- The organization of remote computing resources in different infrastructure levels has improved the computation offloading of robotic applications. For instance, it is preferable to execute delay-sensitive tasks on the robot or offload it to an edge server. While intensive computational tasks would be better to send them to the cloud.
- Mobility is a big challenge for computation offloading for ground and flying robots. To solve this, we have to find the best answer for three questions related to offloading decisions: (1.) why offloading (optimization metric), (2.) where to offload (computing resources), and (3.) how to offload (decision approaches).
- Optimization techniques like heuristics, meta-heuristics, and game theory, are the best choice to find an optimal solution for the computation problem. However, deep learning techniques can learn robots taking adequate offloading decisions. The combination of both techniques would be very promising.

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.

References

[1] A. Viguria, I. Maza, A. Ollero, Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions, Adv. Robot. 24 (1–2) (2010) 1–23.

[2] D. Chatziparaschis, M.G. Lagoudakis, P. Partsinevelos, Aerial and ground robot collaboration for autonomous mapping in search and rescue missions, Drones 4 (4) (2020) URL: https://www.mdpi.com/2504-446X/4/4/79, http://dx.doi.org/10.3390/drones4040079.

[3] O. Cheikhrouhou, I. Khoufi, A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy, Comp. Sci. Rev. 40 (2021) 100369.

[4] M. Mammarella, L. Comba, A. Biglia, F. Dabbene, P. Gay, Cooperative agricultural operations of aerial and ground unmanned vehicles, in: 2020 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor, IEEE, 2020, pp. 224–229.

[5] J. Zhang, R. Liu, K. Yin, Z. Wang, M. Gui, S. Chen, Intelligent collaborative localization among air-ground robots for industrial environment perception, IEEE Trans. Ind. Electron. 66 (12) (2018) 9673–9681.

[6] R. Käslin, P. Fankhauser, E. Stumm, Z. Taylor, E. Mueggler, J. Delmerico, D. Scaramuzza, R. Siegwart, M. Hutter, Collaborative localization of aerial and ground robots through elevation maps, in: 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR, IEEE, 2016, pp. 284–290.

[7] H. Huang, A.V. Savkin, Towards the internet of flying robots: A survey, Sensors 18 (11) (2018) 4038.

[8] L. Wang, D. Cheng, F. Gao, F. Cai, J. Guo, M. Lin, S. Shen, A collaborative aerial-ground robotic system for fast exploration, in: International Symposium on Experimental Robotics, Springer, 2018, pp. 59–71.

[9] T.H. Nam, J.H. Shim, Y.I. Cho, A 2.5 D map-based mobile robot localization via cooperation of aerial and ground robots, Sensors 17 (12) (2017) 2730.

[10] R.D. Arnold, H. Yamaguchi, T. Tanaka, Search and rescue with autonomous flying robots through behavior-based cooperative intelligence, J. Int. Humanit. Action 3 (1) (2018) 1–18.

[11] R. Chaâri, F. Ellouze, A. Koubâa, B. Qureshi, N. Pereira, H. Youssef, E. Tovar, Cyber-physical systems clouds: A survey, Comput. Netw. 108 (2016) 260–278, http://www.sciencedirect.com/science/article/pii/S1389128616302699, http://dx.doi.org/10.1016/j.comnet.2016.08.017.

[12] A. Bhattacharya, P. De, A survey of adaptation techniques in computation offloading, J. Netw. Comput. Appl. 78 (2016) http://dx.doi.org/10.1016/j.jnca.2016.10.023.

[13] L. Feng, W. Li, Y. Lin, L. Zhu, S. Guo, Z. Zhen, Joint computation offloading and URLLC resource allocation for collaborative MEC assisted cellular-V2X networks, IEEE Access 8 (2020) 24914–24926.

[14] A. Boukerche, V. Soto, Computation offloading and retrieval for vehicular edge computing: Algorithms, models, and classification, ACM Comput. Surv. 53 (4) (2020) 1–35.

[15] G. Hu, W.P. Tay, Y. Wen, Cloud robotics: architecture, challenges and applications, IEEE Netw. 26 (3) (2012) 21–28.

[16] R. Arumugam, V.R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F.F. Kong, A.S. Kumar, K.D. Meng, G.W. Kit, DAvinCi: A cloud computing framework for service robots, in: Robotics and Automation (ICRA), 2010 IEEE International Conference on, 2010, pp. 3084–3089, http://dx.doi.org/10.1109/ROBOT.2010.5509469.

[17] L. Qingqing, F. Yuhong, J.P. Queralta, T.N. Gia, H. Tenhunen, Z. Zou, T. Westerlund, Edge computing for mobile robots: multi-robot feature-based lidar odometry with FPGAs, in: 2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network, ICMU, IEEE, 2019, pp. 1–2.

[18] N. Mohamed, J. Al-Jaroodi, I. Jawhar, Utilizing fog computing for multi-robot systems, in: 2018 Second IEEE International Conference on Robotic Computing, IRC, IEEE, 2018, pp. 102–105.

[19] M. Raue, S.G. Scholl, The Use of Heuristics in Decision Making under Risk and Uncertainty, Springer International Publishing, Cham, 2018 pp. 153–179, URL: http://dx.doi.org/10.1007/978-3-319-92478-6_7 http://dx.doi.org/10.1007/978-3-319-92478-6_7.

[20] K. Hussain, M. Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. (2018) http://dx.doi.org/10.1007/s10462-017-9605-z.

[21] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38, http://dx.doi.org/10.1109/MSP.2017.2743240.

[22] M. Sohrabi, H. Azgomi, A survey on the combined use of optimization methods and game theory, Arch. Comput. Methods Eng. (2018) http://dx.doi.org/10.1007/s11831-018-9300-5.

[23] S. Lee, D. Har, D. Kum, Drone-assisted disaster management: Finding victims via infrared camera and lidar sensor fusion, in: 2016 3rd Asia-Pacific World Congress on Computer Science and Engineering, APWC on CSE, IEEE, 2016, pp. 84–89.

[24] L. Hawley, W. Suleiman, Control framework for cooperative object transportation by two humanoid robots, Robot. Auton. Syst. 115 (2019) 1–16.

[25] M.S. Kaiser, S. Al Mamun, M. Mahmud, M.H. Tania, Healthcare robots to combat COVID-19, in: COVID-19: Prediction, Decision-Making, and Its Impacts, Springer, 2021, pp. 83–97.

[26] S. Deshmukh, R. Shah, Computation offloading frameworks in mobile cloud computing : a survey, in: 2016 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC, 2016, pp. 1–5.

[27] G. Carvalho, B. Cabral, V. Pereira, J. Bernardino, Computation offloading in edge computing environments using artificial intelligence techniques, Eng. Appl. Artif. Intell. 95 (2020) 103840, URL: http://www.sciencedirect.com/science/article/pii/S0952197620302050, http://dx.doi.org/10.1016/j.engappai.2020.103840.

[28] A. Shakarami, M. Ghobaei-Arani, M. Masdari, M. Hosseinzadeh, A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective, J. Grid Comput. 18 (4) (2020) 639–671.

[29] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1628–1656.

[30] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective, Comput. Netw. 182 (2020) 107496, URL: http://www.sciencedirect.com/science/article/pii/S1389128620311634, http://dx.doi.org/10.1016/j.comnet.2020.107496.

[31] L. Lin, X. Liao, H. Jin, P. Li, Computation offloading toward edge computing, Proc. IEEE 107 (8) (2019) 1584–1607.

[32] C. Jiang, X. Cheng, H. Gao, X. Zhou, J. Wan, Toward computation offloading in edge computing: A survey, IEEE Access 7 (2019) 131543–131558.

[33] A. Shakarami, A. Shahidinejad, M. Ghobaei-Arani, A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective, Softw. - Pract. Exp. 50 (9) (2020) 1719–1759.

[34] S. Dubey, J. Meena, Computation offloading techniques in mobile edge computing environment: A review, in: 2020 International Conference on Communication and Signal Processing, ICCSP, IEEE, 2020, pp. 1217–1223.

[35] M. Ghobaei-Arani, A. Souri, A.A. Rahmanian, Resource management approaches in fog computing: a comprehensive review, J. Grid Comput. 18 (1) (2020) 1–42.

[36] H. Wu, Multi-objective decision-making for mobile cloud offloading: A survey, IEEE Access 6 (2018) 3962–3976.

[37] S. Kumar, M. Tyagi, A. Khanna, V. Fore, A survey of mobile computation offloading: Applications, approaches and challenges, in: 2018 International Conference on Advances in Computing and Communication Engineering, ICACCE, 2018, pp. 51–58.

[38] K. Akherfi, M. Gerndt, H. Harroud, Mobile cloud computing for computation offloading: Issues and challenges, Appl. Comput. Inform. 14 (1) (2018) 1–16, URL: http://www.sciencedirect.com/science/article/pii/S2210832716300400, http://dx.doi.org/10.1016/j.aci.2016.11.002.

[39] L. Liu, Y. Du, Q. Fan, W. Zhang, A survey on computation offloading in the mobile cloud computing environment, Int. J. Comput. Appl. Technol. 59 (2019) 106, http://dx.doi.org/10.1504/IJCAT.2019.098031.

[40] A.M. Rahmani, M. Mohammadi, A.H. Mohammed, S.H.T. Karim, M.K. Majeed, M. Masdari, M. Hosseinzadeh, Towards data and computation offloading in mobile cloud computing: Taxonomy, overview and future directions, Wirel. Pers. Commun. (2021) 1–39.

[41] F. Rubio, F. Valero, C. Llopis-Albert, A review of mobile robots: Concepts, methods, theoretical framework, and applications, Int. J. Adv. Robot. Syst. 16 (2) (2019) 1729881419839596.

[42] M.A. Khan, A survey of computation offloading strategies for performance improvement of applications running on mobile devices, J. Netw. Comput. Appl. 56 (2015) 28–40.

[43] S. Mu, Z. Zhong, D. Zhao, M. Ni, Joint job partitioning and collaborative computation offloading for internet of things, IEEE Internet Things J. 6 (1) (2018) 1046–1059.

[44] S.H. Seo, J. Straub, Comparative analysis of graph partitioning algorithms in context of computation offloading, in: 2017 IEEE International Conference on Electro Information Technology, EIT, IEEE, 2017, pp. 1–7.

[45] G. Li, Q. Lin, J. Wu, Y. Zhang, J. Yan, Dynamic computation offloading based on graph partitioning in mobile edge computing, IEEE Access 7 (2019) 185131–185139.

[46] M. Geist, B. Scherrer, O. Pietquin, A theory of regularized markov decision processes, in: International Conference on Machine Learning, PMLR, 2019, pp. 2160–2169.

[47] M.E. Mkiramweni, C. Yang, J. Li, Z. Han, Game-theoretic approaches for wireless communications with unmanned aerial vehicles, IEEE Wirel. Commun. 25 (6) (2018) 104–112.

[48] A. Rahman, J. Jin, A. Rahman, A. Cricenti, M. Afrin, Y. ning Dong, Energy-efficient optimal task offloading in cloud networked multi-robot systems, Comput. Netw. 160 (2019) 11–32, URL: http://www.sciencedirect.com/science/article/pii/S1389128619306371, http://dx.doi.org/10.1016/j.comnet.2019.05.016.

[49] C. Gao, V.C.S. Lee, Energy efficient mobile computation offloading through workload migration, in: 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), 2015, pp. 1147–1150, http://dx.doi.org/10.1109/SmartCity.2015.225.

[50] H.Q. Le, H. Al-Shatri, A. Klein, Efficient resource allocation in mobile-edge computation offloading: Completion time minimization, in: 2017 IEEE International Symposium on Information Theory, ISIT, 2017 pp. 2513–2517, http://dx.doi.org/10.1109/ISIT.2017.8006982.

[51] M. Afrin, J. Jin, A. Rahman, Y.-C. Tian, A. Kulkarni, Multi-objective resource allocation for edge cloud based robotic workflow in smart factory, Future Gener. Comput. Syst. 97 (2019) 119–130, URL: http://www.sciencedirect.com/science/article/pii/S0167739X18326785, http://dx.doi.org/10.1016/j.future.2019.02.062.

[52] A. Yousefpour, G. Ishigaki, R. Gour, J.P. Jue, On reducing IoT service delay via fog offloading, IEEE Internet Things J. 5 (2) (2018) 998–1010, http://dx.doi.org/10.1109/JIOT.2017.2788802.

[53] H. Qian, D. Andresen, Extending mobile device's battery life by offloading computation to cloud, in: 2015 2nd ACM International Conference on Mobile Software Engineering and Systems, 2015, pp. 150–151, http://dx.doi.org/10.1109/MobileSoft.2015.39.

[54] S. Melendez, M.P. McGarry, Computation offloading decisions for reducing completion time, in: 2017 14th IEEE Annual Consumer Communications Networking Conference, CCNC, 2017, pp. 160–164, http://dx.doi.org/10.1109/CCNC.2017.7983099.

[55] S. Melendez, M.P. McGarry, Computation offloading decisions for reducing completion time, in: 2017 14th IEEE Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2017, pp. 160–164.

[56] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, Z. Liu, A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading, IEEE Trans. Intell. Transp. Syst. 22 (6) (2020) 3664–3674.

[57] B. Huang, Y. Li, Z. Li, L. Pan, S. Wang, Y. Xu, H. Hu, Security and cost-aware computation offloading via deep reinforcement learning in mobile edge computing, Wirel. Commun. Mob. Comput. 2019 (2019).

[58] M.T.M. Emmerich, A.H. Deutz, A tutorial on multiobjective optimization: fundamentals and evolutionary methods, Nat. Comput. 17 (3) (2018) 585–609, URL: http://dx.doi.org/10.1007/s11047-018-9685-y, http://dx.doi.org/10.1007/s11047-018-9685-y.

[59] C. Von Lücken, B. Baran, C. Brizuela, A survey on multi-objective evolutionary algorithms for many-objective problems, Comput. Optim. Appl. (2014) 1–50, http://dx.doi.org/10.1007/s10589-014-9644-1.

[60] K. Deb, K. Deb, Multi-Objective Optimization, Springer US, Boston, MA, 2014, pp. 403–449, URL: http://dx.doi.org/10.1007/978-1-4614-6940-7-15 http://dx.doi.org/10.1007/978-1-4614-6940-7_15.

[61] T. Besedeš, C. Deck, S. Sarangi, M. Shor, Age effects and heuristics in decision making, Rev. Econ. Stat. 94 (2) (2012) 580–595, URL: http://dx.doi.org/10.1162/REST_a_00174, http://dx.doi.org/10.1162/REST_a_00174, arXiv: http://dx.doi.org/10.1162/REST_a_00174.

[62] S. Voß, Meta-heuristics: The state of the art, in: A. Nareyek (Ed.), Local Search for Planning and Scheduling, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 1–23.

[63] J. Qiu, Q. Wu, G. Ding, Y. Xu, S. Feng, A survey of machine learning for big data processing, EURASIP J. Adv. Signal Process. 2016 (2016) http://dx.doi.org/10.1186/s13634-016-0355-x.

[64] S.S. Mousavi, M. Schukat, E. Howley, Deep reinforcement learning: an overview, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2016, pp. 426–440.

[65] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117, URL: http://www.sciencedirect.com/science/article/pii/S0893608014002135, http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[66] M.E. Mkiramweni, C. Yang, J. Li, W. Zhang, A survey of game theory in unmanned aerial vehicles communications, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3386–3416.

[67] A.B. PI, M. Blum, M. Kearns, T. Sandholm, M. Hajiaghayi, Machine Learning, Game Theory, and Mechanism Design for a Networked World.

[68] S. Mahjoubi, Y. Bao, Game theory-based metaheuristics for structural design optimization, Comput.-Aided Civ. Infrastruct. Eng. (2021).

[69] H. Firdaus, S. Hassan, H. Kaur, A comparative survey of machine learning and meta-heuristic optimization algorithms for sustainable and smart healthcare, Afr. J. Comput. ICT 11 (4) (2018).

[70] Z. Tian, S. Fong, Survey of Meta-Heuristic Algorithms for Deep Learning Training, 2016, http://dx.doi.org/10.5772/63785.

[71] O. Saha, P. Dasgupta, A comprehensive survey of recent trends in cloud robotics architectures and applications, Robotics 7 (3) (2018) URL: https://www.mdpi.com/2218-6581/7/3/47, http://dx.doi.org/10.3390/robotics7030047.

[72] O. Saha, R. Dasgupta, A comprehensive survey of recent trends in cloud robotics architectures and applications, Robotics 7 (2018) http://dx.doi.org/10.3390/robotics7030047.

[73] M. Alhamad, T. Dillon, E. Chang, A survey on SLA and performance measurement in cloud computing, in: R. Meersman, T. Dillon, P. Herrero, A. Kumar, M. Reichert, L. Qing, B.-C. Ooi, E. Damiani, D.C. Schmidt, J. White, M. Hauswirth, P. Hitzler, M. Mohania (Eds.), On the Move to Meaningful Internet Systems: OTM 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 469–477.

[74] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.

[75] O. Cheikhrouhou, A. Koubâa, A. Zarrad, A cloud based disaster management system, J. Sens. Actuator Netw. 9 (1) (2020) 6.

[76] V. Ovidiu, F. Peter, Internet of things: Converging technologies for smart environments and integrated ecosystems, in: River Publishers Series in Communication, 2013.

[77] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, Mobile edge computing: Survey and research outlook, 2017, arXiv preprint arXiv:1701.01090.

[78] A. Koubaa, A. Ammar, M. Alahdab, A. Kanhouch, A.T. Azar, Deepbrain: Experimental evaluation of cloud-based computation offloading and edge computing in the internet-of-drones for deep learning applications, MDPI Sens. 20 (18) (2020) http://dx.doi.org/10.3390/s20185240.

[79] E. Ahmed, M.H. Rehmani, Mobile edge computing: Opportunities, solutions, and challenges, Future Gener. Comput. Syst. 70 (2017) 59–63, URL: http://www.sciencedirect.com/science/article/pii/S0167739X16303260, http://dx.doi.org/10.1016/j.future.2016.09.015.

[80] K. Dolui, S.K. Datta, Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing, in: 2017 Global Internet of Things Summit, GIoTS, 2017, pp. 1–6.

[81] Y. Jararweh, L. Tawalbeh, F. Ababneh, A. Khreishah, F. Dosari, Scalable cloudlet-based mobile computing model, Procedia Comput. Sci. 34 (2014) 434–441, URL: http://www.sciencedirect.com/science/article/pii/S1877050914009065, http://dx.doi.org/10.1016/j.procs.2014.07.051, The 9th International Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.

[82] F. Bonomi, R. Milito, Fog computing and its role in the internet of things, in: Proceedings of the MCC Workshop on Mobile Cloud Computing, 2012, http://dx.doi.org/10.1145/2342509.2342513.

[83] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 37–42, URL: http://dx.doi.org/10.1145/2757384.2757397, http://dx.doi.org/10.1145/2757384.2757397.

[84] Y.C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile Edge Computing a Key Technology Towards 5G, Technical Report ETSI White Paper No. 11, ETSI, Sophia Antipolis CEDEX, France, 2015, p. 16.

[85] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, C. Lin, Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment, IEEE Access 6 (2018) 1706–1717.

[86] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, M. Shoaib, Bringing computation closer toward the user network: Is edge computing the solution? IEEE Commun. Mag. 55 (11) (2017) 138–144.

[87] H. Bangui, S. Rakrak, S. Raghay, B. Buhnova, Moving to the edge-cloud-of-things: Recent advances and future research directions, 7 (2018) 309, http://dx.doi.org/10.3390/electronics7110309.

[88] H.E. Ciritoglu, T. Saber, T.S. Buda, J. Murphy, C. Thorpe, Towards a better replica management for Hadoop distributed file system, in: 2018 IEEE International Congress on Big Data (BigData Congress), 2018, pp. 104–111.

[89] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, vol. 3, 2009.

[90] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari, et al., Understanding service-oriented architecture (SOA): A systematic literature review and directions for further investigation, Inf. Syst. 91 (2020) 101491.

[91] R. Doriya, P. Chakraborty, G.C. Nandi, Robotic services in cloud computing paradigm, in: Cloud and Services Computing (ISCOS), 2012 International Symposium on, 2012, pp. 80–83, http://dx.doi.org/10.1109/ISCOS.2012.24.

[92] I.A.T. Hashem, N.B. Anuar, A. Gani, I. Yaqoob, F. Xia, S.U. Khan, MapReduce: Review and open challenges, Scientometrics 109 (1) (2016) 389–422, URL: http://dx.doi.org/10.1007/s11192-016-1945-y, http://dx.doi.org/10.1007/s11192-016-1945-y.

[93] L. Turnbull, B. Samanta, Cloud robotics: Formation control of a multi robot system utilizing cloud infrastructure, in: Southeastcon, 2013 Proceedings of IEEE, 2013, pp. 1–4, http://dx.doi.org/10.1109/SECON.2013.6567422.

[94] L. Bingwei, C. Yu, B. Erik, P. Khanh, S. Dan, C. Genshe, A holistic cloud-enabled robotics system for real-time video tracking application, in: Future Information Technology, Lecture Notes in Electrical Engineering, vol. 276, Springer Berlin Heidelberg, 2014, pp. 455–468.

[95] G. Mohanarajah, D. Hunziker, R. D'Andrea, M. Waibel, Rapyuta: A cloud robotics platform, IEEE Trans. Autom. Sci. Eng. 12 (2) (2015) 481–493, http://dx.doi.org/10.1109/TASE.2014.2329556.

[96] Z. Yin, J. Liu, C. Zhao, X. Ge, A cloud architecture for service robots, in: International Conference on Cooperative Design, Visualization and Engineering, Springer, 2018, pp. 49–56.

[97] R. Chaari, O. Cheikhrouhou, A. Koubâa, H. Youssef, H. Hmam, Towards a distributed computation offloading architecture for cloud robotics, in: 2019 15th International Wireless Communications Mobile Computing Conference, IWCMC, 2019, pp. 434–441, http://dx.doi.org/10.1109/IWCMC.2019.8766504.

[98] P. Le Noac'h, A. Costan, L. Bougé, A performance evaluation of Apache Kafka in support of big data streaming applications, in: 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 4803–4806.

[99] M. Iqbal, T. Soomro, Big data analysis: Apache storm perspective, Int. J. Comput. Trends Technol. 19 (2015) 9–14, http://dx.doi.org/10.14445/22312803/IJCTT-V19P103.

[100] V.K. Sarker, J. Peña Queralta, T.N. Gia, H. Tenhunen, T. Westerlund, Offloading SLAM for indoor mobile robots with edge-fog-cloud computing, in: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology, ICASERT, 2019, pp. 1–6.

[101] D. Spatharakis, M. Avgeris, N. Athanasopoulos, D. Dechouniotis, S. Papavassiliou, A switching offloading mechanism for path planning and localization in robotic applications, in: 2020 International Conferences on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), IEEE, 2020, pp. 77–84.

[102] A. Koubâa, B. Qureshi, M.-F. Sriti, A. Allouch, Y. Javed, M. Alajlan, O. Cheikhrouhou, M. Khalgui, E. Tovar, Dronemap planner: A service-oriented cloud-based management system for the internet-of-drones, Ad Hoc Netw. 86 (2019) 46–62, URL: http://www.sciencedirect.com/science/article/pii/S1570870518306814, http://dx.doi.org/10.1016/j.adhoc.2018.09.013.

[103] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, M. Khalgui, Micro air vehicle link (MAVlink) in a nutshell: A survey, IEEE Access 7 (2019) 87658–87680.

[104] W.-S. Jung, J. Yim, Y.-B. Ko, S. Singh, Acods: adaptive computation offloading for drone surveillance system, in: 2017 16th Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net, IEEE, 2017, pp. 1–6.

[105] C. Qu, S. Wang, P. Calyam, Dycoco: A dynamic computation offloading and control framework for drone video analytics, in: 2019 IEEE 27th International Conference on Network Protocols, ICNP, IEEE, 2019, pp. 1–2.

[106] D. Chemodanov, C. Qu, O. Opeoluwa, S. Wang, P. Calyam, Policy-based function-centric computation offloading for real-time drone video analytics, in: 2019 IEEE International Symposium on Local and Metropolitan Area Networks, LANMAN, IEEE, 2019, pp. 1–6.

[107] M. Berman, J.S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, GENI: A federated testbed for innovative network experiments, Comput. Netw. 61 (2014) 5–23.

[108] A.M. Rahmani, T.N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach, Future Gener. Comput. Syst. 78 (2018) 641–658.

[109] T.N. Gia, L. Qingqing, J.P. Queralta, Z. Zou, H. Tenhunen, T. Westerlund, Edge AI in smart farming IoT: CNNs at the edge and fog computing with LoRa, in: 2019 IEEE AFRICON, IEEE, 2019, pp. 1–6.

[110] R. Wang, Y. Cao, A. Noor, T.A. Alamoudi, R. Nour, Agent-enabled task offloading in UAV-aided mobile edge computing, Comput. Commun. 149 (2020) 324–331.

[111] S. Luo, H. Li, Z. Wen, B. Qian, G. Morgan, A. Longo, O. Rana, R. Ranjan, Blockchain-based task offloading in drone-aided mobile edge computing, IEEE Netw. 35 (1) (2021) 124–129.

[112] X. Qiu, L. Liu, W. Chen, Z. Hong, Z. Zheng, Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing, IEEE Trans. Veh. Technol. 68 (8) (2019) 8050–8062.

[113] A. Rahman, J. Jin, A. Cricenti, A. Rahman, D. Yuan, A cloud robotics framework of optimal task offloading for smart city applications, in: 2016 IEEE Global Communications Conference, GLOBECOM, 2016, pp. 1–7.

[114] M. Gen, L. Lin, Genetic Algorithms, American Cancer Society, 2008, pp. 1–15, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470050118.ecse169 http://dx.doi.org/10.1002/9780470050118.ecse169, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470050118.ecse169.

[115] A. Rahman, J. Jin, A.L. Cricenti, A. Rahman, A. Kulkarni, Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance, IEEE Trans. Ind. Inf. 15 (5) (2019) 2500–2511.

[116] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, K. Nakamura, QoS-aware robotic streaming workflow allocation in cloud robotics systems, IEEE Trans. Serv. Comput. (2018).
[117] Y. Wang, N. Zhang, Z. Zhuo, C. Kang, D. Kirschen, Mixed-integer linear programming-based optimal configuration planning for energy hub: Starting from scratch, Appl. Energy 210 (2018) 1141–1150.
[118] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[119] Y. Guo, Z. Mi, Y. Yang, M.S. Obaidat, An energy sensitive computation offloading strategy in cloud robotic network based on GA, IEEE Syst. J. 13 (3) (2019) 3513–3523.
[120] Y. Guo, Z. Mi, Y. Yang, J. Yan, M. Obaidat, An energy sensitive system framework for cloud robotic network, Int. J. Commun. Syst. 32 (14) (2019) e4028, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4028, http://dx.doi.org/10.1002/dac.4028, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.4028, e4028 dac.4028.
[121] S. Khuller, M. Purohit, K.K. Sarpatwar, Analyzing the optimal neighborhood: Algorithms for partial and budgeted connected dominating set problems, SIAM J. Discrete Math. 34 (1) (2020) 251–270, URL: http://dx.doi.org/10.1137/18M1212094, http://dx.doi.org/10.1137/18M1212094, arXiv:http://dx.doi.org/10.1137/18M1212094.
[122] M. Noto, H. Sato, A method for the shortest path search by extended Dijkstra algorithm, in: Smc 2000 Conference Proceedings. 2000 Ieee International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions' (Cat. No.0), vol. 3, 2000, pp. 2316–2320, vol.3.
[123] S. Chinchali, A. Sharma, J. Harrison, A. Elhafsi, D. Kang, E. Pergament, E. Cidon, S. Katti, M. Pavone, Network offloading policies for cloud robotics: a learning-based approach, 2019, ArXiv abs/1902.05703.
[124] Z. Wei, J. Xu, Y. Lan, J. Guo, X. Cheng, Reinforcement learning to rank with Markov decision process, 2017, pp. 945–948, http://dx.doi.org/10.1145/3077136.3080685.
[125] Z. Hong, H. Huang, S. Guo, W. Chen, Z. Zheng, QoS-aware cooperative computation offloading for robot swarms in cloud robotics, IEEE Trans. Veh. Technol. 68 (4) (2019) 4027–4041.
[126] F. Xu, W. Yang, H. Li, Computation offloading algorithm for cloud robot based on improved game theory, Comput. Electr. Eng. 87 (2020) 106764, URL: http://www.sciencedirect.com/science/article/pii/S0045790620306194, http://dx.doi.org/10.1016/j.compeleceng.2020.106764.
[127] O. Sefraoui, M. Aissaoui, M. Eleuldj, Openstack: Toward an open-source solution for cloud computing, Int. J. Comput. Appl. 55 (2012) 38–42, http://dx.doi.org/10.5120/8738-2991.
[128] K.M. Kumaran, M. Chinnadurai, Cloud-based robotic system for crowd control in smart cities using hybrid intelligent generic algorithm, J. Ambient Intell. Humaniz. Comput. (2020) 1–14.
[129] A. Shahidinejad, M. Ghobaei-Arani, Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach, Softw. - Pract. Exp. 50 (12) (2020) 2212–2230.
[130] M. Penmetcha, B.-C. Min, A deep reinforcement learning-based dynamic computational offloading method for cloud robotics, IEEE Access (2021).
[131] A. Shakarami, A. Shahidinejad, M. Ghobaei-Arani, An autonomous computation offloading strategy in mobile edge computing: A deep learning-based hybrid approach, J. Netw. Comput. Appl. 178 (2021) 102974.
[132] X. Wang, H. Guo, Mobility-aware computation offloading for swarm robotics using deep reinforcement learning, in: 2021 IEEE 18th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2021, pp. 1–4.
[133] F. Jazayeri, A. Shahidinejad, M. Ghobaei-Arani, Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach, J. Ambient Intell. Humaniz. Comput. 12 (8) (2021) 8265–8284.
[134] F. Farahbakhsh, A. Shahidinejad, M. Ghobaei-Arani, Multiuser context-aware computation offloading in mobile edge computing based on Bayesian learning automata, Trans. Emerg. Telecommun. Technol. 32 (1) (2021) e4127.
[135] F. Farahbakhsh, A. Shahidinejad, M. Ghobaei-Arani, Context-aware computation offloading for mobile edge computing, J. Ambient Intell. Humaniz. Comput. (2021) 1–13.
[136] K. Jo, J. An, J. Jung, H. Min, An offloading decision scheme for a multi-drone system, in: 17th IIE International Conference on Computer, Electrical, Electronics and Communication Engineering, 2017, pp. 1–6.
[137] R. Valentino, W.-S. Jung, Y.-B. Ko, A design and simulation of the opportunistic computation offloading with learning-based prediction for unmanned aerial vehicle (uav) clustering networks, Sensors 18 (11) (2018) 3751.
[138] X. Chen, Z. Ji, Y. Fan, Y. Zhan, Restful API architecture based on laravel framework, J. Phys.: Conf. Ser. 910 (1) (2017) 012016, IOP Publishing.
[139] J. Yao, N. Ansari, Online task allocation and flying control in fog-aided internet of drones, IEEE Trans. Veh. Technol. 69 (5) (2020) 5562–5569.
[140] Y. Chu, F. You, Integrated planning, scheduling, and dynamic optimization for batch processes: MINLP model formulation and efficient solution methods via surrogate modeling, Ind. Eng. Chem. Res. 53 (34) (2014) 13391–13411.
[141] B. Kim, J. Jung, H. Min, J. Heo, Energy efficient and real-time remote sensing in AI-powered drone, Mob. Inf. Syst. 2021 (2021).
[142] X.-W. Chen, X. Lin, Big data deep learning: challenges and perspectives, IEEE Access 2 (2014) 514–525.
[143] F. Jahan, W. Sun, Q. Niyaz, M. Alam, Security modeling of autonomous systems: A survey, ACM Comput. Surv. 52 (2019) 1–34, http://dx.doi.org/10.1145/3337791.
[144] A. Allouch, O. Cheikhrouhou, A. Koubâa, K. Toumi, M. Khalgui, T. Nguyen Gia, Utm-chain: blockchain-based secure unmanned traffic management for internet of drones, Sensors 21 (9) (2021) 3049.
[145] O. Cheikhrouhou, Secure group communication in wireless sensor networks: a survey, J. Netw. Comput. Appl. 61 (2016) 115–132.
[146] S. Jain, R. Doriya, Security Issues and Solutions in Cloud Robotics: A Survey, 2019, pp. 64–76, http://dx.doi.org/10.1007/978-981-15-1718-1_6.
[147] A. Allouch, O. Cheikhrouhou, A. Koubâa, M. Khalgui, T. Abbes, MAVSec: Securing the mavlink protocol for ardupilot/PX4 unmanned aerial systems, in: 2019 15th International Wireless Communications & Mobile Computing Conference, IWCMC, IEEE, 2019, pp. 621–628.
[148] C. Karri, O. Cheikhrouhou, A. Harbaoui, A. Zaguia, H. Hamam, Privacy preserving face recognition in cloud robotics: A comparative study, Appl. Sci. 11 (14) (2021) 6522.
[149] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A.M. Zanchettin, S. Zanero, An experimental security analysis of an industrial robot controller, in: 2017 IEEE Symposium on Security and Privacy, SP, 2017, pp. 268–286.
[150] M.I. Tariq, N.A. Memon, S. Ahmed, S. Tayyaba, M.T. Mushtaq, N.A. Mian, M. Imran, M.W. Ashraf, A review of deep learning security and privacy defensive techniques, Mob. Inf. Syst. 2020 (2020).
[151] C. Briggs, Z. Fan, P. Andras, A review of privacy-preserving federated learning for the internet-of-things, 2020, arXiv e-prints, arXiv–2004.
[152] P.P. Ray, An introduction to dew computing: Definition, concept and implications, IEEE Access 6 (2017) 723–737.
[153] A. Botta, L. Gallo, G. Ventre, Cloud, fog, and dew robotics: Architectures for next generation applications, in: 2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), IEEE, 2019, pp. 16–23.