Future Generation Computer Systems 154 (2024) 251–265

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Review Article

Research allocation in mobile volunteer computing system: Taxonomy, challenges and future work

Peizhe Ma [a,*], Saurabh Garg [a], Mutaz Barika [b]

[a] University of Tasmania, Churchill Ave, Hobart, 7005, TAS, Australia
[b] Newcastle University , 1 Science Square, Newcastle, NE4 5TG, Newcastle upon Tyne, United Kingdom

ARTICLE INFO

Keywords:
Distributed computing
Resource management
Edge computing
Cloud computing
Volunteer computing
IoT

ABSTRACT

The rise of mobile devices and the Internet of Things has generated vast data which require efficient processing methods. Volunteer Computing (VC) is a distributed network that utilises idle resources from diverse devices for task completion. VC offers a cost-effective and scalable solution for computation resources. Mobile Volunteer Computing (MVC) capitalises on the abundance of mobile devices as participants. However, managing a large number of participants in the network presents a challenge in scheduling resources. Various resource allocation algorithms and MVC platforms have been developed, but there is a lack of survey papers summarising these systems and algorithms. This paper aims to bridge the gap by delivering a comprehensive survey of MVC, including related technologies, MVC architecture, and major finding in taxonomy of resource allocation in MVC.

1. Introduction

The Internet of Things (IoT) enables communication and information exchange between physical objects, such as self-driving cars, through the internet [1]. This technology has greatly improved existing computing networks and quality of life. With the increasing amount of data generated by IoT devices, there is growing interest in extracting valuable information from this data. Various technologies, including cloud, edge computing, and volunteer computing, have been developed to handle IoT data [2–5]. Volunteer Computing (VC) is a distributed computing system that leverages crowdfunding to use idle resources from participating devices for computationally intensive tasks in a cost-effective manner. VC breaks down projects into smaller chunks, providing affordable, scalable, and powerful computing power.

In a typical VC platform, millions of users contribute their idle resources, such as memory and processing power [6]. For example, Uber Driving serves as a real-world example of VC. When a person needs a ride, they request a task through the Uber application. The application acts as a central service, assigning the task to an available and capable driver who uses their own idle resource, their car, to provide the ride. The key distinction is that in VC, participants donate their idle resources, while in the case of Uber, users pay to use the application.

With the rise of mobile devices, which had an estimated 1.6 billion users in 2013, computing power has become more accessible to individuals [7]. Mobile volunteer computing (MVC), a type of VC where all participants are mobile devices, takes advantage of unique features of mobile devices such as mobility and low cost. There exists a growing trend in MVC, but there is a lack of surveys focusing on high-level system structure and resource allocation in MVC.

In particular, a small number of survey papers related to crowd computing have focused MVC, with only 11 papers including discussions of MVC technologies [8–18]. However, these 11 papers tend to concentrate on either sensors or application systems rather than resource management. Only two of the papers briefly mention resource management as one of the important challenges in the current industry, but they lack a comprehensive taxonomy of resource management techniques used by the current industry.

As a result, this paper aims to address the need for future research and fill in the gap in this area by investigating the following research questions in this article: (1) What are different technologies for crowd computing and their differences? (2) What are the general structures and systems for MVC?, and (3) What are current resource allocation technologies for MVC?.

To solve the above resource questions and meet future needs, this article makes the following contributions:

- A detailed taxonomy of mobile volunteer computing (MVC) with a specific focus on crowd computing (CC) and volunteering computing (VC) using mobile devices

* Corresponding author.
E-mail address: peizhe.ma@utas.edu.au (P. Ma).

https://doi.org/10.1016/j.future.2024.01.015
Received 15 July 2023; Received in revised form 16 December 2023; Accepted 10 January 2024
Available online 17 January 2024

- Discuss the architecture and famous MVC systems from processing-based and sensing-based perspectives.
- Propose a resource allocation taxonomy in MVC environment from the perspectives of application, target resource and scheduling policy.
- Identify current open research issues in MVC systems and resource allocation algorithms based on the result of our scoping review.

Based on the best of our knowledge, this paper is the first of its kind to analyse both the taxonomies of MVC and resource allocation. The rest of the paper is organised as follows. Section 2 reviews the background for this survey paper, including the research strategy and related survey. Section 3 MVC taxonomy is discussed. Following this, Section 4 outlines the concept of MVC and the system taxonomy. Section 5 provides a taxonomy of classifications of MVC resource allocation. Section 6 illustrates the future directions based on this survey paper and Section 7 concludes this paper.

2. Background

2.1. Research methodology

As this paper needs to examine emerging evidence in a specific topic — a taxonomy of VC systems and resource allocation in VC, the scoping review is selected compared with the systematic review. A scoring review focuses on the general characteristics of certain topics, while a systematic literature review summarises solutions to a specific question [19]. Using the scoping review, an overview of a broad topic can be presented within a short period of time [20]. As a result, the scoping review is conducted to learn about the concepts of volunteer computing and resource allocation policies. Research keywords are defined based on the project topic including 'edge computing', 'fog computing', 'volunteer computing', 'mobile crowd computing', 'desktop computing', 'Internet of Things (IoT) applications', 'edge computing resource allocation scheduling', 'fog computing resource allocation scheduling', 'volunteer computing resource allocation scheduling', and 'crowd computing resource allocation scheduling'. A scoping review is conducted using Google Scholar, IEEE Xplore, Science Direct and Springer, since they includes a range of research articles, particularly in computer science and offers a convenient way to search for literature across multiple academic sources. Research papers for scheduling algorithms and techniques are limited to 2014 to 2022 as the concept of volunteer computing roughly came about around 2014. A set of protocols is defined as the inclusion criteria to filter the research articles shown below:

- The focus of the article is on resource allocation/scheduling in VC/MVC environment
- Peer-reviewed publications
- Research articles written in the English language.

Our exclusion criteria are as follows:

- Research articles written in languages other than English.
- The duplicate of the same article retrieved by different databases.

After applying our selection criteria to the survey papers about resource management policies in Mobile Volunteer Computing, we have identified a total of 73 papers that align with our criteria. Moving forward, these papers will form the foundation for our primary objective: presenting the taxonomy. The mind mapping in Fig. 1 visually represents the research process, while the detailed taxonomy is discussed in Section 5.

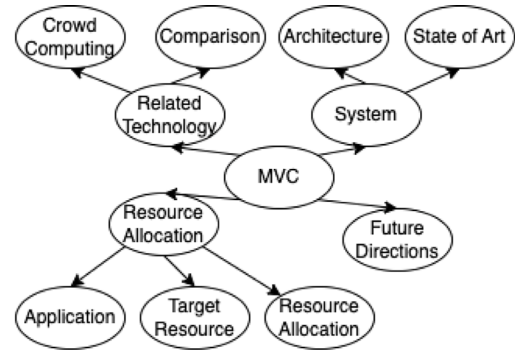

Fig. 1. Mind-mapping diagram.

2.2. Related surveys

Table 1 summarises 28 survey papers that exhibit similarities to our research. These papers are collected based on the keywords 'Volunteer Computing Survey', 'Crowd Computing', 'Mobile Computing Survey', 'Mobile Crowd Computing Survey', and 'Mobile Cloud Computing Survey'. In addition, the current study delves into the concept of MVC and categorises MVC systems based on sensors and applications. The aforementioned table features three columns that indicate whether or not the surveyed papers discuss MVC and MVC systems. Moreover, since this paper also includes a taxonomy of classifications of resource scheduling, an additional column has been incorporated at the end of the table.

In general, 12 of the surveyed papers have referenced the concept of MVC. However, not all of them have incorporated a system taxonomy. For instance, Boubiche et al. (2019) only classify MCC based on application types such as environment and social [12]. On the other hand, Ali et al. (2021) include sensor technology as one of the categories when categorising Mobile Crowding Sensing (MCS) for traffic management systems but ignoring the applications [17]. Out of these papers, only three have covered both the aspects of VC system and resource management. Abualsaud et al. (2018) discussed MCC systems from the perspective of sensing groups and applications, but did not mention anything related to resource management [10]. Boubiche et al. (2019) and Capponi et al. (2019) are the only two papers that discussed all four areas, including MVC, sensing, applications, and resource management [12,13]. However, Boubiche et al. (2019) only briefly mentioned resource allocation as one of the challenges for MCC and did not provide a detailed classification [12]. Similarly, Capponi et al. (2019) only stated that resource allocation is one of the targets of the MCC process [13].

To summarise, based on the review of existing survey papers, none of them have given due attention to the system side of MVC and resource allocation in the industry. Hence, this paper aims to fill this gap by covering both these aspects through a detailed analysis of MVC taxonomy and resource allocation classification.

3. Landscape of related technologies

3.1. Crowd computing

The concept of crowd computing is a relatively recent development with early references dating back to the early 2000s, and most of the research being conducted from 2009 onwards [38]. In simple terms, crowd computing can be defined as the process of solving problems by utilising crowdsourcing, which leverages the collective wisdom of the crowd. Several authors have defined the concept of crowd computing from various perspectives. Murray et al. (2010) seem to be the first authors that introduce crowd computing as networks designed to spread

Table 1
Related survey comparison.

Articles	Mobile volunteer	Sensors	Application	Resource management
Choi et al. (2007) [21]	✗	✗	✓	✗
Yuen, King, and Leung (2011) [22]	✗	✗	✓	✗
Enzai and Tang (2014) [23]	✗	✗	✗	✓
La and Kim (2014) [24]	✗	✗	✗	✓
Ahmed et al. (2015) [25]	✗	✗	✓	✓
Guo et al. (2015) [8]	✓	✗	✓	✗
Jaimes, Vergara-Laurens, and Raij (2015) [9]	✓	✗	✗	✗
Liu et al. (2015) [26]	✗	✗	✗	✓
Marosi and Lovas (2015) [27]	✗	✗	✗	✓
Wang et al. (2015) [28]	✗	✗	✓	✓
Zare et al. (2015) [29]	✗	✗	✗	✓
Paranjothi et al. (2017) [30]	✗	✗	✗	✓
Abualsaud et al. (2018) [10]	✓	✓	✓	✗
Gu et al. (2018) [31]	✗	✗	✗	✓
Noor et al. (2018) [32]	✗	✗	✓	✗
Wang et al. (2018) [11]	✓	✗	✓	✗
Zhou and Rajkumar (2018) [33]	✗	✗	✗	✓
Boubiche et al. (2019) [12]	✓	✓	✓	✓
Capponi et al. (2019) [13]	✓	✓	✓	✓
Mengistu and Che (2019) [14]	✓	✗	✗	✓
Vahdat-Nejad et al. (2019) [15]	✓	✗	✗	✗
Aliyu et al. (2020) [34]	✗	✗	✓	✓
Shamshirband et al. (2020) [35]	✗	✗	✓	✗
Waheed et al. (2020) [16]	✓	✗	✓	✗
Ali et al. (2021) [17]	✓	✓	✗	✗
Rahmani et al. (2021) [36]	✗	✗	✗	✓
Sisi and Souri (2021) [18]	✓	✗	✓	✓
Maray and Shuja (2022) [37]	✗	✗	✓	✓

computation and aggregate results [39]. Miller et al. (2010) propose crowd computing as the human analogue to the cloud, combining the strengths of both cloud computing and human [40]. Schneider et al. (2012) provide an umbrella term for crowd computing as a myriad of human interaction tools that enable individuals to exchange ideas, make decisions and utilisation of the world's resources [41]. Particularly, Muhammadi and Rabiee (2013) cited Wikipedia as one of the most well-known examples of the crowdsourcing system [42]. Millions of internet users contribute and create this power-free online encyclopedia. Additionally, crowd computing can be classified into different application classes based on its characteristics, such as Web 2.0 and social computing, crowdsourcing, human computation, and audience/crowd computer interaction [38,41]. Given that this paper will concentrate on mobile volunteer computing, the following sections will specifically propose some related sub-classes of crowd computing, including desktop computing, crowdsensing, and volunteer computing.

3.1.1. Desktop computing

Desktop Computing (DC) is a form of distributed computing that utilises idle computing resources on the Internet to run computationally expensive projects over a low-speed network. Participating individuals are typically unused desktop or laptop computer users who have agreed to donate their resources [43–45]. DC uses only free resources and does not interfere with users' primary activities [46]. The idea of DC was first introduced in 1987 by Litzkow (1987) [47]. In 1999, Sarmenta and Hirano (1999) further discussed the concept of Volunteer Computing (VC), which enhances the computational power of DC by utilising VC [48]. DC is based on two main pillars: computational pillars for managing and configuring DC, and participative pillars for attracting more participants [43]. According to Choi et al. (2004), DC consists of three major components: the client who requests the task, volunteers who are idle desktops or laptops, and volunteer servers that manage volunteers [44]. DC technology are also used by different popular systems. For example, BOINC (Berkeley Open Infrastructure for Network Computing) is one of the famous open-source platforms by using DC technology and helps maintain several remarkable scientific results such as the Einstein@home project [49]. Fig. 2 is summarised based on the analysis above, particularly based on the idea from [21]. DC is categorised based on four dimensions, including centralised and distributed organisation, web-based platform and middleware-based, internet and local area network scale, as well as voluntary and non-voluntary resource providers. Based on these features, DC provides several benefits, including [50]:

- Cost-effectiveness: DC is relatively cheaper to run projects compared to other computing systems.
- Scalability: DC can scale to thousands of computers, which makes it a suitable option for computationally expensive tasks.
- Ease of deployment and support: DC is easy to set up and requires minimal maintenance, which makes it a practical option for organisations with limited resources.

Despite its benefits, DC also faces several challenges, which include:

- Limited resources: DC's computing power is limited by the number of available resources, such as the number of participating computers and their processing capabilities.
- Slow connection: DC's performance can be affected by the slow internet connection of some participants, which can result in project delays.
- Lack of trust: DC relies on the trust of participants, which can be challenging to establish and maintain.
- Heterogeneity: The heterogeneity of software and hardware used by participants can lead to compatibility issues, which can complicate project deployment and management.

3.1.2. Mobile crowd sensing

Traditionally, different sets of sensors have been deployed at fixed locations to collect and transmit data. However, with the rapid development of smart devices such as smartphones, a majority of individuals now own and carry them wherever they go. In light of this, crowd sensing has emerged as an alternative solution for enhancing the capabilities of sensing. This methodology distributes the sampling of phenomena of interest across a large number of individuals [9,51]. Given that many of these individuals are mobile device users, this approach is also referred to as mobile crowd sensing. Based on its name, we can also discuss its characteristics from three aspects [52]:

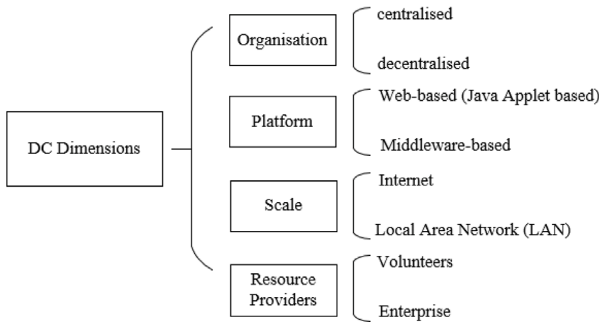

Fig. 2. A taxonomy of DC dimensions.

- Mobile: The resources utilised in mobile crowd sensing include user-owned devices such as smartphones, wearable devices, tablets, smart vehicles, and so on [13,51]. Given that these devices are typically carried or worn by individuals, they are likely to be in moving status during data collection, enabling a large coverage area. However, controlling the quality of the data collected from these devices can be challenging.
- Crowd: As a subclass of crowd computing, mobile crowd sensing exhibits the defining characteristic of leveraging the collective wisdom of the public to achieve its objectives [13]. A wide range of smart devices can be incorporated into the network to contribute to the sensing task. However, a major feature of this approach is the inherent heterogeneity of the participating devices, which can lead to diversity in the types and quality of sensing data.
- Sensing: In mobile crowd sensing, the majority of participants are typically assigned with sensing-related tasks, as opposed to computational tasks. For instance, one of the real-world applications of this approach is noise mapping, which is a significant problem in urban areas [53]. To monitor noise levels, a noise map is generated to assess the level of pollution. Mobile crowd sensing techniques can be employed in this context by using smartphones to collect noise data along with location information, which can be used to update the noise map.

Participants can join the mobile crowd sensing network using various communication technologies. Capponi et al. (2019) provide a taxonomy of mobile crowd sensing based on its communication technologies [13]. This network is categorised into two main groups, namely infrastructured and infrastructure-less. Infrastructured communication relies on base stations or access points, such as cellular networks and wireless local area networks. In contrast, infrastructure-less communication is a peer-to-peer technology in which devices can directly communicate with each other using protocols such as WiFi, LTE-Direct, and Bluetooth. Mobile crowd sensing is a multi-layer structure. Capponi et al. (2019) separate them into application, data, communication, sensing layers [13]. Guo et al. (2014) also propose a similar structure for mobile crowd sensing including application, data processing, data collection, and crowd sensing [51]. Mobile crowd sensing offers several advantages, including:

- Mobility: Due to the mobility of participants and their devices, mobile crowd sensing can provide higher coverage and collect data from various locations [13].
- Scalability: As a form of crowd computing, mobile crowd sensing is highly scalable and suitable for different types of projects. The scales of mobile crowd sensing can be characterised into three categories: group, community, and urban, according to the size and scope of the sensing project [8].
- Low cost: The sensing task can be divided into small sensing chunks, which reduces the cost of data collection and processing [8].
- Use of human knowledge: Mobile crowd sensing can leverage the intelligence of human participants to generate more contextual and accurate data. For example, humans can easily identify free parking spots and submit information into the system [13].

Mobile crowd sensing also has several limitations, including:

- Incentive methodologies: The success of mobile crowd sensing relies on the participation of a large number of individuals, but it can be difficult to find effective incentive methodologies to attract and retain participants [9,13].
- Heterogeneity: Mobile crowd sensing also has the limitation of heterogeneity, where different types of smart devices may generate data in different formats, making it difficult to manage and analyse the data together [13,54].
- Personal preferences: As individuals are involved in the process, they naturally have personal preferences which might not align with the initial goal of the sensing project [54].
- Security: Security and privacy are important limitations in mobile crowd sensing, as individuals' devices are used in the network and their security and privacy need to be considered. Additionally, when aggregating results from different participants, data integrity should also be checked [54].

In addition, there is a concept called "crowdsourcing" that can cause confusion among people. Both mobile crowd sensing and mobile crowdsourcing are based on crowd computing, which involves distributing tasks to a large group of participants. Mobile crowd sensing employs mobile devices and sensors to collect data and complete environmental tasks, such as creating an air pollution map, without the involvement of humans in most cases. On the other hand, mobile crowdsourcing involves humans as the main stakeholders [55]. When a task publisher publishes a task, the central server distributes it to individual crowd workers to complete and submit the task. There are two main types of mobile crowdsourcing applications: human intelligence and human sensor [56]. Human intelligence uses human knowledge, which is easy for humans to solve but difficult for computers, such as natural language processing. The human sensor uses humans as sensors to collect data based on their observations apart from sensors on mobile devices, such as the smart transportation system.

3.1.3. Volunteer computing

Volunteer Computing (VC) is a network-based distributed computing system, which utilises idle resources from participating devices to complete some computational expensive programs and has gained recognition under various names such as public fog. Heterogeneity is one of the important characteristics of the VC system as a broad range of devices can act as volunteer nodes. These nodes typically include but are not limited to desktops, mobiles, tablets, network gateways, and smart TVs that have heterogeneous hardware, operating systems, and versions [4,14]. VC system consists of two entities which are resources and controllers and can be classified into three categories: centralised, decentralised and hybrid [14,57]. A centralised system, also referred to as client/server, emphasises the existence of server machine(s) for control. A decentralised system is also known as a peer-to-peer system (P2P), in which each node can communicate with the others and serve as a resource and controller simultaneously. A hybrid structure merges the features of the previous two structures, allowing nodes to communicate with each other while maintaining a global directory. Additionally, VC uses master-worker parallel computing to complete tasks [6,58]. Specifically, the master receives the task and divides it into sub-tasks, which are then assigned to different workers. Workers perform calculations for sub-tasks and transfer the results back to the master, who validates and combines them. As for the master, it can work as both a controller and a worker. Generally, VC systems are classified by operating environment, including volunteer grid, volunteer cloud, volunteer mobile, and parallel volunteer computing [14,58]. There are several advantages provided by VC:

- Scalability: VC is highly scalable as millions of volunteer nodes can connect to the system at the same time [6,14]. For example, one of the most famous VC platforms, Berkeley Open Infrastructure of Network Computing (BOINC), has the capacity to manage 1.1 million volunteer nodes [14,58].
- Cost-effective: As a distributed computing system, there is no cost associated with purchasing a large amount of hardware to support computationally expensive projects. Instead, all participants volunteer to join the network and the cost can be reduced [6].
- Speed: VC can accelerate the computation process by breaking down the entire task into smaller chunks. Each participant can work on a specific chunk in parallel with others to reduce the overall time needed to complete the task.
- Accessibility: The nature of VC allows for broad accessibility, as individuals and organisations can easily join the network through the internet and utilise its resources.

Aside from some advantages mentioned above, several noticeable issues also remain open in volunteer computing that should be addressed.

- Resource allocation: Due to the heterogeneity feature mentioned above, volunteer nodes have a diverse range of hardware and software. Moreover, the availability time and resources for these nodes can change dynamically. Therefore, it is difficult to identify patterns and group these devices to efficiently allocate resources. Tapparello et al. (2016) discuss that an efficient algorithm is needed to utilise resources and achieve the best possible results through distributed computation [59].
- Incentive models: Most end devices participate in VC systems due to their interest in or sense of belonging [14]. Useful incentive models are critical to attracting more devices to join the system, expand the capacity of the system, and perform more tasks.
- Result aggregation and validation: VC systems operate in master-worker parallel computing in which the master divides tasks among workers to perform calculations and send them back to the master for aggregation and validation [6]. However, validation of the results consumes a large number of computing resources. Furthermore, it is not easy to aggregate and validate results from heterogeneous nodes.
- Security: A secure environment is crucial to implement tasks using idle devices. To ensure the security and privacy of these idle devices during their participation in the VC system, as well as the transformation between devices, a security mechanism must be in place.

Resource allocation is selected as the primary target for this study to categorise VC from the four open questions discussed above, since it is the first thing to plan when setting up a VC environment. Efficient resource allocation is crucial for setting up a VC environment as it enables the system to determine available resources and plan their usage to achieve the project's goals.

3.1.4. Comparative analysis of volunteer computing with other paradigms

Based on our analysis in the previous subsection, Table 2 summarises the differences between the three technologies including desktop computing, mobile crowd sensing, and volunteer computing. Although they are all part of the crowd computing paradigm, their specific characteristics vary. The differences are categorised into four aspects as follows:

- Purpose: The purpose of each technology is different. Desktop computing is mainly used for personal and business applications, while mobile crowd sensing is used for data sensing and collection, particularly for monitoring purposes. Volunteer computing is typically utilised for computationally expensive tasks and scientific research projects.
- Resource: The differences in the types of participants for these technologies are reflected in their names. Desktop computing uses unused desktops or laptops. However, the participants in mobile crowd sensing and volunteer computing are more varied. Mobile crowd sensing uses user-owned devices such as mobile phones and wearable devices, while volunteer computing can use any type of device such as phones and computers.
- Connection: Desktop computing relies on low-speed network connections, while mobile crowd sensing and volunteer computing can leverage any available communication methods such as cellular networks and WiFi.
- Computation Power: Desktop computing and mobile crowd sensing mainly rely on individual devices' power to run the task or sense the data. Volunteer computing combines resources from different devices to provide a high-quality platform to run tasks.

3.2. Comparison between VC, edge computing and cloud

In addition to comparing the three subclasses of crowd computing, it is also worthwhile to distinguish volunteer computing from edge computing and cloud computing, as individuals often confuse them. Edge computing aims to bring computational and storage resources physically close to the end users, as well as reduce transformation cost and response time [60]. It enables applications to access network information in real time while maintaining low latency. Cloud computing can rapidly deliver resources to users based on their demands through the Internet [61]. Generally, cloud computing can be classified into three service models, which are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), as well as four deployment models including private, public, community and hybrid.

Table 3 compares volunteer computing, edge computing, and cloud computing with a focus on resource allocation. Despite being situated at different levels of the network, their resource locations differ from their physical locations. Volunteer and cloud computing derive resources from their respective levels, while edge computing sources from all levels, as noted by [60]. Volunteer and edge computing, positioned closer to end devices, exhibit lower network latency than cloud computing. Notably, cloud computing provides unlimited on-demand resources, billed according to usage, while volunteer and edge computing face limitations in computational resources, storage, and power. Edge computing adopts a pricing strategy similar to the cloud, while volunteer computing, relying on donated resources from idle devices, offers relatively cheaper rates. In volunteer computing, the donating devices own the resources, setting it apart from edge and cloud computing.

4. Mobile volunteer computing system

Mobile Volunteer Computing (MVC) is a type of distributed system that shares the same characteristics as Volunteer Computing (VC). The primary difference between the two is the type of participating devices. While VC can include any device, such as computers, servers, and mobile devices, MVC focuses specifically on mobile devices like phones and tablets. The mobility and human intelligence of the MVC network are improved due to the easy portability and user involvement of mobile devices. Additionally, mobile devices have powerful features like cameras and GPS, which make them ideal for diverse tasks like sensing. However, MVC may require more participants to complete large projects, as the computational power of mobile devices is relatively limited compared to computers. Nonetheless, due to the increasing usage and rapid development of mobile devices, it is essential to research and focus on MVC. Therefore, this paper will discuss the MVC architecture and system taxonomy in detail in this section.

Table 2
Desktop computing vs. Mobile crowd sensing vs. Volunteer computing.

Characteristics	Desktop computing	Mobile crowd sensing	Volunteer computing
Purpose	Personal or business applications	Data sensing and collection	Computationally expensive tasks and scientific projects
Resource	Unused desktop or laptop computer users	User-companioned devices (mobile phones, wearable devices, and smart vehicles, smart cards, and so on)	Any type of devices
Connection	Low-speed network	Infrastructured and infrastructure-less	Any communication methods
Computation power	Individual devices' power	Individual devices' sensors	Computing platform

Table 3
Summary of comparison between Volunteer, Edge and Cloud.

Characteristics	Volunteer	Edge	Cloud
Size/Scalability	1,000,000,000s [14]	1000s	100s to 1000s [62]
Capacity	Limited resources and storage	Limited resources and storage [63]	On demand provision
Major components level	End device level	Edge level	Cloud level
Resource location	End device level	All levels [60]	Cloud level
Resource management	Distributed	Distributed [64]	Centralised [64]
Resource ownership	Multiple	Single	Single [61]
Node operating system	Multiple	Multiple	Multiple
Network	Low latency	Low latency	High latency
Battery	Limit	Limit	Not limit
Mobility	Movable	Movable	Steady
Pricing	Cheaper	Based on the resources	Based on the resources

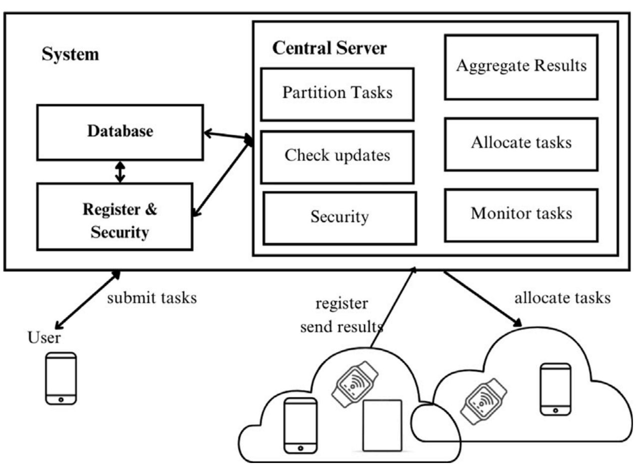

Fig. 3. MVC high-level architecture.

4.1. System architecture

In this paper, we have collected and examined 23 well-known mobile-related crowd-computing systems from the literature. After our analysis, the high-level architecture of MVC systems is presented in Fig. 3. This architecture consists of three primary components: users who submit tasks, mobile devices that act as volunteers, and the central MVC system. Our focus here is on the central system, which includes seven major functions:

- Register and Security: The register function receives information from both mobile devices and users, recording device availability times and capacity, as well as information on submitted tasks such as data size and time requirements. In some of the systems, it can issue proof of certification to ensure network integrity, particularly for security concerns. The registration authority of the AnonySense system, for example, issues authenticity to its scheduling server and aggregation server for participant verification [65]. Therefore, the security function is discussed with the register function. The Register function can be part of the central server or a separate component within the system, such as the Testbed authority component in CrownLab, or the control server in CANDIS that includes all functions, including registering devices [66,67].
- Database: The database function is simply used to store the data collected from the register, such as information about participants and tasks. Therefore, when the central server needs to distribute tasks, it will retrieve the required data from the database. Additionally, the database can be updated by the central server in case of any changes occurring in the current network. For example, in the designed architecture of GEMCloud, there are only two components within the system: the server and the database, where the server requests to update the database and receive the updated version of information from the client when it receives information from the devices [66].
- Partition Tasks: The partition tasks serve as splitting the tasks submitted by the user into small chunks based on the current network capability. However, not all of these 23 MVC systems include the partition tasks function. ANGELS is the one that considers partitioning tasks [68]. Its designed framework has a component called the partition which will consider the capability of current resources from the resource pool and then split the tasks accordingly.
- Allocate Tasks: The allocation function is a critical component in all MVC systems as it is responsible for managing and distributing tasks to different participants based on various scenarios. Each platform has its unique scheduling algorithm. For instance, Ocelot has two servers, namely the database and scheduling server, with the latter being responsible for allocating tasks to different devices to ensure that user-submitted jobs are completed within the specified time [69].
- Check updates and monitor Tasks: The functions of checking updates and monitoring tasks are often combined to track the progress of ongoing tasks and update the relevant information in the database. This function facilitates communication between the central server and the devices, allowing for changes such as updating device availability times. Not all systems, however, discuss these functions. For instance, the jUniGrid system includes a job monitoring component to monitor the status of devices and tasks [70].
- Aggregate results: The aggregation function is an essential component of MVC systems, responsible for collecting and combining

results from individual devices and returning them to the user. Nearly all 23 MVC systems in this study include an aggregation function. For instance, CANDIS employs a central server that performs all functions, including aggregating results [67].

4.2. Current state of the art in MVC platforms

Based on the aforementioned architecture, we categorise the MVC system into two types: processing-based and sensing-based. Processing-based MVC systems are designed to handle large computational tasks, whereas sensing-based MVC systems primarily collect and sense data, such as environmental data in a particular region. A categorisation of 23 well-known mobile-related crowd computing systems has been performed based on two categories: seven sensing-based systems (AnonySense [65], PRISM [71], Mobiscope [72], LiveCompare [73], Microblog [74], Bubble-sensing [75], AirShower@home [76]) and 15 processing-based systems (GEMCloud [77], CWC [78], Serendipity [79], FemtoClouds [80], Mobile Device Clouds [81], CellCloud [82], Hyrax [83], Unity [84], Mobile OSGI.NET [85], jUniGrid [70], Ocelot [69], CANDIS [67], ANGELS [68], Honeybee [86], DRAP [87]). These statistics reveal that the majority of these systems are processing-based and intended to handle computation-related tasks. However, CrowdLab is the only system that emphasises both processing and sensing aspects [66]. There are six systems among the 23 mobile-related crowd computing systems that specifically target MVC (CrowdLab [66], AirShower@home [76], CWC [78], Serendipity [79], Ocelot [69], DRAP [87]). The details for each of these six systems will be discussed in the below paragraphs:

CrowdLab: The first testbed architecture utilising volunteer mobile resources is CrowdLab [66]. It integrates with existing infrastructure-based testbeds without removing any functions. CrowdLab achieves this by running virtual mobiles over volunteer nodes, protecting both data and locations using isolating guest codes and decentralising testbed services. The architecture comprises Testbed Authorities (TAs) and Device Authorities (DAs). TAs, centralised components, handle experiment registration, and resource distribution, and provide a root of trust. DAs, decentralised, connect to single devices for resource control. A geographic grouping of DAs is a site, communicating through ad-hoc networks. During experiments, CrowdLab can run processing-based tasks, including measurement and sensing tasks.

AirShower@home: The study by Gordienko et al. (2015) focuses on the idea of transitioning from passive volunteer computing to other forms of volunteer actions, such as measurement [76]. They highlight a specific project called AirShower@home, which aims to solve the scientific problem of estimating the possibility of identifying air showers in the solar system and the frequency and distribution of air showers in different cities to create a virtual online map. The measurements are carried out using volunteers' tools such as camera chips and GPS. The workflow of AirShower@home consists of three parts, namely, volunteers participating in the network, mobile applications used to detect flashes and send data, and servers used to process data and create an online map. The system has four components, including hardware (various gadgets), brainware (human actions), software (mobile applications), and the volunteer community.

Computing While Charging (CWC): Arslan et al. (2012) presented a framework named CWC, which utilises mobile devices for distributed computing [78]. The framework is designed to utilise mobile devices provided by the company to perform tasks. The tasks will only be executed when the mobile devices are charging. If the owner of the device disconnects from a power outlet, the task will be suspended and migrated to another charged device. The system design includes a scheduler that minimises makespan by taking into account both CPU and bandwidth, and an efficient migration approach for task migration.

Serendipity: In their work, Shi et al. (2012) proposed a processing-based architecture named Serendipity, which aims to assist a mobile device in executing computationally intensive tasks that surpass its own capabilities by utilising other mobile devices to minimise local power consumption and computation time [79]. Serendipity is designed as a decentralised structure, where an individual mobile device node consists of three different components: a job engineer, a master, and several worker processes, that communicate with others and request resources. The general process begins with the job engine, which receives tasks from users to construct a task profile and allocates tasks to workers by initiating a job. Once workers complete tasks, the master aggregates results and returns them back to the job engine.

Ocelot: Xu et al. (2013) proposed Ocelot, a distributed mobile computing platform based on Berkeley Open Infrastructure for Network Computing (BOINC) [69]. Ocelot is designed to solve highly parallel and lightweight computational tasks in order to conserve energy resources. Unlike BOINC, which uses servers and workstations, Ocelot uses smartphones and tablets to reduce maintenance costs and power usage. Ocelot can also be applied in wireless sensor networks to solve computational tasks. The Ocelot system is divided into two parts: the database server and the scheduling/web/push server. The database server stores different types of information such as user information, device information, and task results. The scheduling/web/push server is the core component of the system, responsible for managing tasks and devices, maintaining the system, handling requests, and performing other related functions.

DRAP: Agarwal and Nayak (2015) proposed DRAP, a decentralised computing system that groups volunteer mobile devices into a cloudlet [87]. DRAP relies entirely on volunteer-based services, which results in reduced costs. Each device in DRAP is considered a part of the cluster and must communicate with other devices and serve as middleware. DRAP consists of three main components: device manager, neighbour discovery, and cloud controls. The device manager detects devices that are suitable for serving requests and being part of the cloudlet. The neighbour discovery component communicates with neighbouring devices to determine if they can join the network and provide services. The cloud controls component provides cloudlet services and distributes tasks. DRAP is based on the general requirements of the cloudlet environment.

In this section, we provide a detailed review of MVC architecture that will be used to investigate different resource allocations in the following section.

5. Taxonomy of resources allocation in MVC

The complexity and need for resource allocation in MVC are significant, as mobile devices have limited resources. An effective allocation policy is crucial for managing resources and achieving the final goal of completing tasks successfully. However, due to the heterogeneity and unstable nature of volunteered resources, it is challenging to develop a resource allocation policy that can manage these resources and balance the workload effectively. Therefore, we present a research taxonomy that aggregates all related works for resource allocation into the application, target resource and scheduling policy perspectives in volunteer and edge paradigms. Fig. 4 shows the high-level structure of our resource allocation taxonomy. This taxonomy will be discussed in detail in the following.

5.1. Application

From the perspective of the application, we further divide it into categories of 'types' and 'users' as demonstrated in Fig. 5. 'Types' refer to what the algorithm is scheduled for and allocated for, including independent tasks, dependent tasks and stream applications. 'Users' represent the number of users considered.

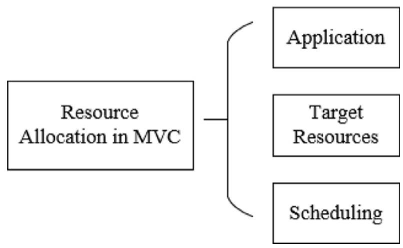

Fig. 4. Resources allocation in MVC.

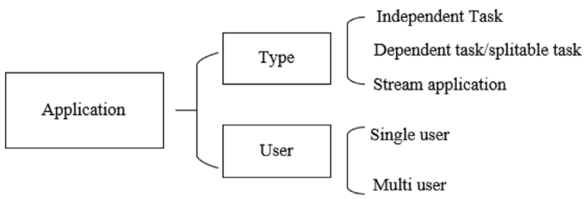

Fig. 5. Application-based classification of resources allocation algorithms.

5.1.1. Type

To design an efficient resource allocation method, it is crucial for the process to know the type and properties of the application and thus understand what it is being allocated for. The type of application can be classified into independent tasks application (or bag-of-tasks application), dependent tasks application and stream application. Research works that discussed the aforementioned types of application will be explained in detail in the following sections:

1. Independent Task
   The tasks are the requests and data that are submitted by end devices to the edge or volunteer computing system. End devices can submit as many tasks as they need to reach their goals. There are two types of tasks, independent tasks and dependent tasks, which are further divided based on their relationships. A task that is independent is one where there is no relationship between the number of tasks submitted by the end devices. The data needed for competing tasks are all provided by the end devices at the beginning. For example, Li et al. (2019) proposed a video processing application composed of independent tasks [88]. The whole video is split into 10–20 MB clips, which corresponds to video processing tasks. All of these tasks will arrive simultaneously at the edge, and they are independent of one another. Based on our analysis for these research works, the majority of research works have focused on the big-of-tasks application with independent tasks.
2. Dependent/Splittable Task
   Dependent/splittable tasks are different from independent tasks because either some computationally intensive tasks can be divided into sub-tasks with dependencies or the application submits dependency-aware tasks. These tasks cannot be run simultaneously as they might request the completion results of certain tasks as input. This subcategory is more realistic compared with the 'independent task' category but only 11 papers address dependencies among tasks [89–99]. Using the assumption of splittable applications, Liu et al. (2020) [89] developed a task scheduling algorithm for a vehicular application, such as augmented vehicular reality which broadens a vehicle's horizontal scope through the sharing of information with nearby vehicles and allocate dependent tasks belonging to the same application [100]. An example of a tourism service application considered by Ni et al. (2017) is the provision of different services, such as transportation and ticketing, with low latency, wide geographical distribution, and high mobility [90]. These applications assume that each task can be further divided into subtasks.
3. Stream Application
   Stream application is that edge or volunteer computing system schedules resources for continuous incoming data. In this scenario, the data arrival rate is a key feature to consider when scheduling. In Zhao et al. (2017) work, the delay-sensitive mobile applications are scheduled and the task arrival process is assumed to be a Poisson process [101]. For example, when simulating, a task's arrival rate is randomly selected from the range of 1 to 30. In Liu et al. (2018) work, streams of blockchain-based video are scheduled with the same time interval between segments [102]. In addition, they set the block size between 0.5 and 2 MB and the generating time between 10 and 20 s.

5.1.2. User

User category divided research works based on the number of users/devices in the end device layer, including a single user and multiple users.

1. Single User
   The single user refers to the fact that there is only one user or device submitting tasks at the end device layer. Due to the rarity of this scenario in real life, only six papers take the single user type into account [90,98,103–106]. Kuang et al. (2019) assumed there is only one mobile user device to submit tasks in the running environment [103].
2. Multi Users
   Apart from the single user, multiple users are commonly considered because many end devices are typically included at the end device layer in real-world scenarios. Both Yuan and Zhou (2020) and Alameddine et al. (2019) considered different types of devices at the end layer, such as phones, tablets and laptops [107,108]. For example, in the experimental setup, Yuan and Zhou (2020) designed five different types of IoT applications [107].

5.2. Target resources

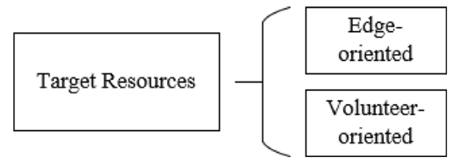

Fig. 6. Target resources based classification of resource allocation.

Under the target resources perspective, there are two categories: edge-oriented and volunteer-oriented, as shown in Fig. 6. Edge-oriented resource allocation in MVC uses edge nodes as the main task processing resources, while volunteer-oriented resource allocation involves volunteer devices participating in the network to help complete tasks. All the research papers that have been collected and reviewed by this study have been categorised based on the two categories of edge-oriented and volunteer-oriented resource allocation. These papers have also been analysed and discussed in detail in the subsequent sections of the study.

5.2.1. Edge-oriented

In the current industry, edge nodes are the primary resource considered when proposing resource allocation in MVC. Unlike volunteer-oriented resources, edge devices are not donation-based, and the system or central server has more control over these nodes. The entire resources of edge nodes can be used only for processing tasks instead of borrowing parts of resources from volunteer nodes. In addition, edge

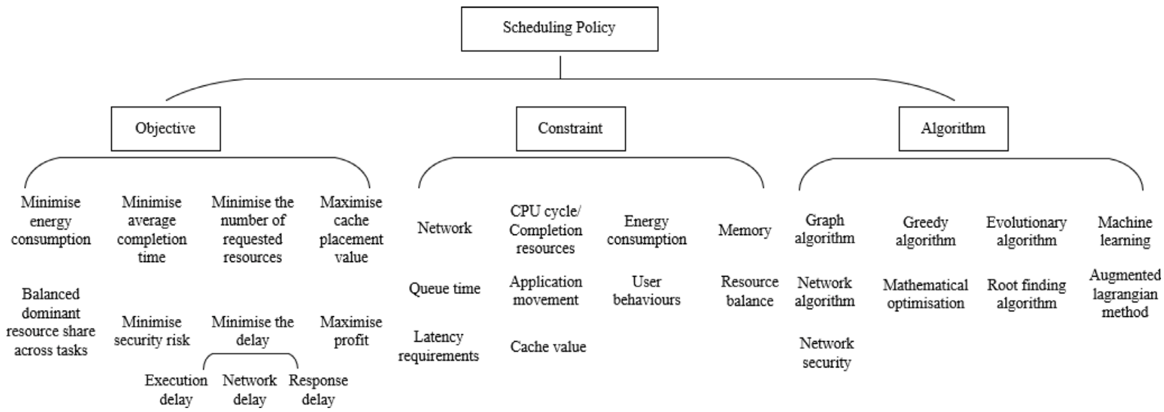

Fig. 7. Policy-based classification of resource allocation.

nodes can provide better security compared to volunteer nodes as the server has full control over nodes and does not need to frequently transfer data when volunteer nodes suddenly become unavailable. For instance, Nath and Wu (2020) proposed a deep reinforcement learning methodology for offloading and scheduling mobile edge resources in a cache-based system environment using machine learning techniques [109]. Li et al. (2019) presented the Time Average Computation Rate Maximisation (TACRM) algorithm to minimise task completion time by utilising mobile edge nodes [110].

5.2.2. Volunteer-oriented

The volunteer-oriented methodology is less focused on the current industry with only eight collected research papers [97,111–117]. However, compared with edge computing, volunteer computing provides high scalability and flexibility. Volunteer computing can be easily scaled up by adding more nodes inside the network to increase overall network capacity. Additionally, as these resources are donated by mobile devices, it is cost-effective compared to the costs of buying edge nodes. Xu et al. (2019) proposed a dynamic task scheduling algorithm that utilises heterogeneous volunteer computing platforms to meet task deadlines [111]. Hoseiny et al. (2021) developed a cost-effective scheduling algorithm by combining both edge and volunteer computing environments [112].

5.3. Resource allocation policy

From the scheduling policy perspective, there are three categories: objective for resource allocation, constraints considered by the application, as well as algorithm types. Additionally, it is also possible for the research works to fall into several types under the same category since they can achieve multiple objectives, consider different constraints, and combine several algorithms during the resource allocation process (see Fig. 7).

5.3.1. Objective

The objective category is further divided into eight subcategories, including energy consumption, average completion time, the number of requested resources, cache placement value, balance dominant resource sharing, security risk, delay and profit.

1. Energy Consumption
According to the analysis for these research works, minimising energy consumption is targeted by the majority of research papers with 27 papers, when proposing their resource allocation algorithms [95,103,105,107–109,117–137]. Due to the nature of end devices and edge devices, battery operations are commonly adopted by them. It is important to schedule as many tasks as possible within the energy limitations. Specifically, those papers discuss the minimisation of energy consumption from two aspects which are the system and the devices. For example, Wang et al. (2019) aimed at reducing the system energy consumption when selecting edge computing and scheduling tasks [126]. Mao, Zhang and Letaief (2017) proposed their algorithm to eliminate the energy consumption for the end devices when offloading tasks to the edge [105].
2. Average Completion Time
TS-IoT applications require tasks to be completed within a limited time frame, otherwise, the results will be meaningless. Therefore, minimising the average completion time of tasks is another essential objective targeted by lots of paper. For example, Yin, Luo and Luo (2018) constructed their task scheduling algorithm to minimise the task completion time [138]. Mukherjee et al. (2019) proposed the strategy to maximise the number of completed tasks within the deadline [139].
3. The Number of Requested Resources
Considering the limited computing and memory resources at the edge, two of the research papers attempt to reduce the number of resources requested while completing tasks on time. A three-layer network architecture is designed by Wang et al.. (2021) [140] in the edge computing satellite network to reduce resource wasting. The authors of [141] applied the deep reinforcement method to reduce the number of requested resources and the average completion time.
4. Cache Placement Value
Maximise cache placement values are considered in two research papers as well. Some edge devices might have certain cache areas located in the memory to reduce transmission costs and improve system performance by fast accessing the cache values. The authors of [142] proposed a cache locality-aware method for better cache values utilisation. Nath and Wu (2020) aimed at reducing the fetch costs of cache contents by designing a machine learning algorithm [109].
5. Balanced Dominant Resources Sharing
The balanced dominant resources sharing among tasks is crucial for resource allocation since allocating excessive resources for one particular task will result in the degradation of the system's performance. However, the fairness issue of task scheduling has only been addressed in one research paper. Bian, Huang and Shao (2019) designed a task scheduling scheme to learn from experience to ensure multi-resource fairness among tasks [96].
6. Security Risk
Security is another important objective to achieve during the resource allocation process. Due to heterogeneity and mobility features for edge nodes/volunteer nodes, as well as the network transmission environment, the security of these nodes and the transmission channels can be easily hacked by others. On the

other hand, only Daoud et al. (2019) recognised the existence of security risks and privacy risks, as well as proposed a trust access control and resource management mechanism to protect the environment [143].

7. Delay
As with the average completion time objective, reducing the delay in the process also addresses the deadline issue for TS-IoT applications. By reducing the delay and responding to requests on time, the total task completion time is reduced. In addition, this category can be divided into three aspects including execution delay, network delay and response delay. Research papers can target one or multiple aspects as well. For example, Wu et al. (2021) design a fuzzy offloading strategy to minimise the execution latency [93]. Ali et al. (2020) introduced a volunteer-supported fog computing environment to minimise network and response delay simultaneously [117].
8. Profit
Economic objectives are discussed in the research papers as some edge computing service providers charge fees for the resources they provide with 12 papers addressing this objective [90,97,102,107,115,124,125,133,144–147]. For example, the authors of [107] designed a profit-maximised collaborative computational offloading strategy to increase the profits of the system. Huang, Li and Chen (2020) [144] discussed the specific objective of maximising edge server revenue.

5.3.2. Constraints

When designing the solutions for the resource allocation problem, lots of constraints are introduced. In particular, ten subcategories under the constraints category are summarised, including network, transmission cost, CPU cycle/computation resources, energy consumption, memory, queue time, application movement, user behaviours, resource balance, latency requirement, and cache value.

1. Network
In the running environment, the data/tasks are transferred to the edge or cloud computing through the network. In the meantime, the results calculated by edge/cloud are transferred back to devices via the Internet. Therefore, the network, including the bandwidth and transmission costs, is an important constraint to consider when designing the architecture with the majority of research papers that focus on this area. Specifically, transmission costs include uploading and/or downloading time duration, the money spent on transferring data, as well as the communication cost among devices and edges. For example, as authors of [133] believe that the data size of the output is much smaller than the input data, they focused primarily on uploading time costs and ignore downloading ones. When Ni et al. (2017) proposed the resource allocation strategy based on priced timed Petri nets, the transmission cost and time constraint are both defined [90].
2. CPU/Computation Resources
CPU/computation resources is the fundamental constraint to be addressed since it is the most basic element needed for the edge to complete the task. Because of this, the most of research papers mention CPUs. In particular, CPU speed or processing speed, computation time and resources all belong to this subcategory. For instance, CPU cycle frequency is addressed by Mao, Zhang and Letaief (2017) when designing the system model [105]. The authors of [148] considered the computing time in their proposed task model.
3. Energy Consumption
Energy consumption constraint also appears to be a common constraint to address since most papers aim to minimise energy consumption as mentioned in Section 5.3.1. Furthermore, since many devices and edge/volunteer nodes need power to operate, considering the energy consumption when designing the system model is more realistic. The battery life is one of the criteria when selecting the manager edge node mentioned in Sun, Lin and Xu (2018) [149]. Power consumption for transferring the data and processing tasks is discussed by Feng et al. (2020) [147].
4. Memory
As the edge/volunteer devices only have limited memory space and computation resources, memory is another essential constraint to be addressed when allocating resources. When designing system architectures, however, memory space is rarely considered. For example, both the authors of [150,151] discussed the storage capacity for the edge servers when formulating the problem.
5. Queue Time
Queue time is discussed in some of the papers since they assume there is a queue subsystem in the edge server [88,95,107,112,122,128,131,135,152]. A task will wait in the queue if there are not enough resources to compute the task at the edge. The authors of [95,135] mentioned the length of the queue and the queueing time when proposing the system model.
6. Application Movement
The movement of applications is one of the major constraints of mobile edge computing. A vehicular edge computing system, in particular, allows mobile devices to be moved around the environment. Dynamic changes are made to the location and distance between a device and an edge. Four papers mention this constraint when proposing their algorithm [89,98,106,153]. For example, Saleem et al. (2020) considered the human mobility feature when offloading tasks [106]. The authors of [153] specifically proposed a vehicular edge computing algorithm by considering vehicular movement.
7. User Behaviours
Only two research papers address user behaviours constraint as they need to categorise users for different purposes [113,143]. Panadero et al. (2017) assigned different labels to the connected nodes, such as low, middle and high, based on their probability of disconnection [113]. Daoud et al. (2019) is the only research work that recognises the existence of security risks and privacy risks [143]. Trust is computed by considering the user's behaviour such as their access history.
8. Resource Balance, Latency Requirement and Cache Value
Resource balance, latency requirement and cache value are explained in the same paragraph as all of them are separately corresponding to one of the scheduling objectives discussed in Section 5.3.1. Resource balance constraint is considered by the same paper when aiming for multi-resource fairness [96]. Some papers address the latency requirement to minimise the delay throughout the process [88,95,107,112,122,128,131,135,152]. For example, Yi, Huang and Cai (2019) proposed the resource management framework considering deadline constraints [133]. Additionally, cache value is discussed when the research work assumes popular tasks or data can be cached on edge servers [109,142,151,154]. The authors of [109] considered cache capacity, size, and locality when allocating resources.

5.3.3. Algorithm

Various types of algorithms are proposed by researchers, including graph algorithm, greedy algorithm, evolutionary algorithm, machine learning, networking algorithm, mathematical optimisation, root finding algorithm, augmented lagrangian method, network security, as well as scheduling algorithm.

1. Graph Algorithm
Graph algorithm consists of a set of vertices that are connected by edges. It can be used for traversing the whole graph to find a specific node or find the path between nodes. Complex

problems can be visualised and organised by using the graph algorithm, but it can be complicated in terms of memory since there are many vertices to handle at the same time. This algorithm is adopted by some research papers as it is a useful tool for searching and matching devices and edge nodes. For example, the authors of [140] proposed a breadth-first-search-based spanning tree algorithm to link devices with edges. An extended Hungarian algorithm is introduced by Wang et al. (2019) to provide an initial matching between resources and data [154].

2. Greedy Algorithm
The greedy algorithm aims to make local optimal choices at each stage without taking the whole picture into account. Therefore, it provides a relatively simple solution to the optimisation problem and some researchers proposed it to achieve a near-optimal solution. However, it only selects the sub-optimal option in the short term, which may end up being the worst long-term choice. Wang et al. (2019) designed an efficient greedy algorithm to obtain task scheduling solutions with less time [126]. The authors of [133] adopted the sub-optimal greedy algorithm to reduce computational complexity.
3. Evolutionary Algorithm
Evolutionary algorithm is the type of algorithm inspired by nature, such as mutation and natural selection, and emulates the behaviours of nature to solve the problem. Any part of the evolutionary algorithm can be tailored and adjusted according to the problem, however, finding a suitable type of evolutionary algorithm is difficult because different parameters provide different results. It can be used for optimisation questions that cannot be solved in polynomial time. For instance, Rafique et al. (2019) combined the particle swarm optimisation and the cat swarm optimisation to efficiently schedule tasks [155]. The genetic algorithm proposed by the authors of [94] is designed to select an efficient scheduling plan.
4. Machine Learning
Machine Learning (ML) focuses on using data and algorithms to train the machine and predict the results without being explicitly programmed. It is an efficient mechanism as it can independently adopt the result based on a large amount of data. Deep reinforcement learning is a relatively common type of ML method used by researchers. On the other hand, training and running the model are time-consuming and resource-intensive. Yang et al. (2018) employed the deep reinforcement learning by adopting an intelligence agent in the edge server to dynamically allocate resources [152]. Additionally, the authors of [93] proposed fuzzy clustering for offloading tasks from TS-IoT applications.
5. Networking Algorithm
Lyapunov optimisation methodology is the only algorithm fall under this category. It optimally controls the dynamic environment by using Lyapunov functions and is an efficient tool for solving the NP-hardness problem. However, in some cases, it is difficult to come up with an appropriate candidate for the Lyapunov function. For instance, both the authors of [123,130] adopted the Lyapunov optimisation techniques to decompose the initial problem into several parts.
6. Mathematical Optimisation
In mathematical optimisation, the best element is selected from several alternatives based on a set of criteria. A set of algorithms under this category are adopted by research papers, including convex optimisation, gradient descent/subgradient, interior point method, and branch and bound. Mathematical optimisation can be used in conjunction with other types of algorithms such as heuristics to combine both strengths but may be computationally expensive as well. For example, Yu, Wang and Guo (2018) proposed convex optimisation for optimising processing capability and Newton algorithm for transmission power [131].
The authors of [152] proposed a deep reinforcement learning for resource allocation and adopt gradient descent to train the parameters.
7. Root Finding Algorithm
Typically, the x where $f(x) = 0$ is referred to as the zeros, which are also called roots. Root finding algorithm is the algorithm to find the zeros. While this algorithm makes finding the root of the equation easy and fast, its rate of convergence is slow, and some types of root finding algorithms cannot guarantee convergence. Two different root-finding algorithms, Newton's approach and the bisection search, are proposed in these works. For example, Newton iterative method is proposed by Wang et al. (2019) for transmission power allocation [128]. The authors of [122] adopted the bisection search to achieve the optimal transmit power and power allocation.
8. Augmented Lagrangian Method
The augmented Lagrangian method addresses nonlinear optimisation problems by minimising augmented Lagrangian functions, involving constraint functions and multipliers at each iteration. However, like the Root Finding algorithm, it can be slow to converge and computationally expensive. Two studies employ the alternate direction method of multipliers (ADMMs) [102, 121], which breaks down convex optimisation into manageable parts. According to [121], ADMMs help determine task offloading decisions. Liu et al. (2018) propose a low-complexity ADMMs-based method for finding optimal solutions in resource scheduling [102].
9. Network Security
Network security methodology is applied by Daoud et al. (2019) as this is the only paper that aims to ensure the security access of edge nodes [143]. They proposed access control management and intrusion detection to monitor the process. These two methodologies can increase the security of the whole environment. However, to prevent incidents, intrusion detection needs security expertise to set the rules, and it is still possible for a hacker to disguise himself as the person who has access to break the access control management.

6. Future directions

The growing interest in MVC and IoT development has led to significant advancements in technology. However, there are still several areas that require further attention and development in the future. This sentence provides an overview of the future directions based on the above section, which will focus on future directions for MVC systems and resource allocation algorithms.

6.1. Mobile volunteer computing

6.1.1. Incentive modules

Challenges: In MVC, participants contribute their idle resources to create a resource pool for computational tasks. Their motivation to join is often driven by personal interests or a sense of community belonging [14]. However, participants can leave if they lose interest, making a large number of participants crucial for the system's success. Sufficient resources from these devices are necessary to execute submitted tasks.

Significance: (1) Increase participants; (2) Utilise resources; (3) Build long term relationship; (4) Building community.

Suggested Solutions: The industry must develop effective incentive modules to retain existing participants and attract more devices to the network. Five directions can be pursued for efficient incentive modules: reputation-based, monetary-based, social-based, game-based, and volunteer-based. Reputation and penalty-based modules can reward devices providing high-quality computations and penalise those with poor quality. Social-based and game-based modules can offer social-related badges or create a gaming system with ranks and challenging

Table 4
Volunteer computing gap analysis.

Articles	User	Type	Objectives	Constraint	Algorithm
Xu et al. (2019) [111]	Multi	Independent	Completion time, Computation delay	CPU, Latency	Greedy
Hoseiny et al. (2021) [112]	Multi	Independent	Completion time, Computation delay	Network, Memory, CPU, Queue	Graph
Zeng et al. (2020) [97]	Multi	Dependent	Computation delay	Network, CPU, Energy	Graph, Evolutionary
Panadero et al. (2017) [113]	Multi	Independent	Computation delay	CPU, User behaviour	Graph, Mathematical optimisation
Panadero et al. (2018) [114]	Multi	Independent	Computation delay	CPU	Mathematical optimisation
Pham et al. (2019) [115]	Multi	Independent	Completion time, Computation delay, Profit	Network, CPU	Mathematical optimisation
Rubab et al. (2015) [116]	Multi	Independent	Computation delay	CPU	Graph
Ali et al. (2020) [117]	Multi	Independent	Energy consumption, Completion time, Response and network delay	CPU	Greedy

tasks. Monetary-based modules can provide monetary rewards, while volunteer-based modules allow participants to donate their resources to specific fields of interest. For example, a participant interested in astronomy can allocate their resources specifically to astronomy-related tasks.

6.1.2. Manage heterogeneity

Challenges: The MVC system accepts different mobile devices to form the system. However, the heterogeneity of mobile devices in the MVC system poses a challenge due to differences in software, operating systems, and hardware [59]. Additionally, they might also have different preferences when joining the network such as availability time.

Significance: (1) Efficiently manage resources; (2) Enhanced scalability; (3) Adaptability to dynamic environment.

Suggested Solutions: It is worthwhile to manage heterogeneity in the system. One of the potential future directions that can be used is to develop a uniform resource platform, such as an application that can be installed on all mobile devices. The platform can then act as an intermediary between the system and individual devices, facilitating communication and resource management.

6.2. Evolution of resource allocation algorithms

Challenges: Based on our taxonomy of resource allocation, this paper identifies four important areas that are neglected and can be considered as the future directions: (a) the volunteer-supported edge environment, (b) task dependencies, (c) memory constraint, (d) resources fairness. In particular, the majority of existing research works in the literature focus on resource allocation in edge computing rather than volunteer computing. Table 4 summarises only eight volunteer computing resource allocation research papers discussed from the perspectives of application, target resource and scheduling policy. Only the authors of [112] considered dependencies among tasks but ignored memory limitation and resource fairness. The authors of [115,117] both proposed a volunteer-supported edge environment but only consider CPU when designing the resource allocation policy. Apart from these, none of these research works argues the importance of fairness among resources in the volunteer network.

Significance: (1) Retentation participants; (2) Resource fairness, particularly in the volunteer environment; (3) Task completion time deduction; (4) Enhancing algorithm reliability to optimise real-world scenarios.

Suggested Solutions: Develop the resource allocation algorithm by considering the four aspects mentioned above and finding the balance among them.

6.3. Enhancing security

Challenges: In the future of technology, security becomes increasingly important as cyber security and data privacy gain significance. Security in mobile crowd computing (MVC) can be divided into two aspects: system security and data transfer security. Ensuring the security of participant devices within the MVC system is a challenge.

Significance: (1) Data and devices protection; (2) Network security; (3) Enhance system dependability; (4) Assure result reliability.

Suggested Solutions: It is crucial to protect these devices from malware and hackers, as well as prevent any malicious activity. Having a contingency plan to respond to potential attacks is essential, and implementing trust management can restrict access to only trusted entities. Additionally, managing the security of data during transfer is vital. Employing data encryption methods can protect the data, but it may impact algorithm efficiency. Striking a balance between safety and efficiency is necessary when enhancing system security.

7. Conclusion

This scoping review explores resource management in MVC, identifying gaps in taxonomy and algorithm/system classification. It examines differences among VC, DC, and MCS and also compares VC, edge computing, and cloud regarding resource ownership. The study proposes a high-level MVC structure and a taxonomy based on 23 mobile-related crowd-computing systems. Specific six MVC systems are discussed. Additionally, a resource allocation taxonomy for MVC is summarised based on application, target resource, and policy perspectives. Future directions include incentive modules, heterogeneity management, resource management, and security.

CRediT authorship contribution statement

Peizhe Ma: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing. **Saurabh Garg:** Formal analysis, Methodology, Supervision, Writing – review & editing. **Mutaz Barika:** Conceptualization, Formal analysis, Supervision, Writing – review & editing.

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.

Data availability

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

References

[1] Korala, et al., Managing time-sensitive IoT applications via dynamic application task distribution and adaptation, Remote Sens. 13 (20) (2021) 4148.

[2] Bibi, et al., Secure distributed mobile volunteer computing with android, ACM Trans. Internet Technol. (TOIT) 22 (1) (2021) 1–21.

[3] Bayliss, et al., Reliability in volunteer computing micro-blogging services, Future Gener. Comput. Syst. 115 (2021) 857–871.

[4] Jauro, et al., Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend, 96 (2020) 106582.

[5] Raabe, et al., Exploring grid computing & volunteer computing: Analyzing daily computing runtimes on the world community grid, Issues Inf. Syst. 21 (3) (2020).

[6] Durrani, et al., Volunteer computing: requirements, challenges, and solutions, J. Netw. Comput. Appl. 39 (2014) 369–380.

[7] Dehlinger, et al., Mobile application software engineering: Challenges and research directions, in: Workshop on Mobile Software Engineering, Vol. 2, 2011, pp. 29–32.

[8] Guo, et al., Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm, ACM Comput. Surv. (CSUR) 48 (1) (2015) 1–31.

[9] Jaimes, et al., A survey of incentive techniques for mobile crowd sensing, IEEE Internet Things J. 2 (5) (2015) 370–380.

[10] Abualsaud, et al., A survey on mobile crowd-sensing and its applications in the IoT era, Ieee Access 7 (2018) 3855–3881.

[11] Wang, et al., Energy saving techniques in mobile crowd sensing: Current state and future opportunities, IEEE Commun. Mag. 56 (5) (2018) 164–169.

[12] Boubiche, et al., Mobile crowd sensing–Taxonomy, applications, challenges, and solutions, Comput. Hum. Behav. 101 (2019) 352–370.

[13] Capponi, et al., A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2419–2465.

[14] Mengistu, et al., Survey and taxonomy of volunteer computing, ACM Comput. Surv. 52 (3) (2019) 1–35.

[15] Vahdat-Nejad, et al., Context-aware computing for mobile crowd sensing: A survey, Future Gener. Comput. Syst. 99 (2019) 321–332.

[16] Waheed, et al., Volunteer computing in connected vehicles: opportunities and challenges, IEEE Netw. 34 (5) (2020) 212–218.

[17] Ali, et al., Traffic efficiency models for urban traffic management using mobile crowd sensing: A survey, Sustainability 13 (23) (2021) 13068.

[18] Sisi, et al., Blockchain technology for energy-aware mobile crowd sensing approaches in Internet of Things, Trans. Emerg. Telecommun. Technol. (2021) e4217.

[19] Munn, et al., Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach, BMC Med. Res. Methodol. 18 (1) (2018) 1–7.

[20] Pham, et al., A scoping review of scoping reviews: advancing the approach and enhancing the consistency, Res. Synth. Methods 5 (4) (2014) 371–385.

[21] Choi, et al., Characterizing and classifying desktop grid, in: Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGrid'07, IEEE, 2007, pp. 743–748.

[22] Yuen, et al., A survey of crowdsourcing systems, in: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, IEEE, 2011, pp. 766–773.

[23] Enzai, et al., A taxonomy of computation offloading in mobile cloud computing, in: 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, IEEE, 2014, pp. 19–28.

[24] La, et al., A taxonomy of offloading in mobile cloud computing, in: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, IEEE, 2014, pp. 147–153.

[25] Ahmed, et al., Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges, J. Netw. Comput. Appl. 52 (2015) 52–68.

[26] Liu, et al., Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions, J. Netw. Comput. Appl. 48 (2015) 99–117.

[27] Marosi, et al., Defining volunteer computing: a formal approach, Comput. Res. Model. 7 (3) (2015) 565–571.

[28] Wang, et al., A survey of mobile cloud computing applications: Perspectives and challenges, Wirel. Pers. Commun. 80 (2015) 1607–1623.

[29] Zare, et al., Resource scheduling in mobile cloud computing: taxonomy and open challenges, in: 2015 IEEE International Conference on Data Science and Data Intensive Systems, IEEE, 2015, pp. 594–603.

[30] Paranjothi, et al., Survey on three components of mobile cloud computing: offloading, distribution and privacy, J. Comput. Commun. 5 (06) (2017) 1.

[31] Gu, et al., Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions, J. Netw. Comput. Appl. 119 (2018) 83–96.

[32] Noor, et al., Mobile cloud computing: Challenges and future research directions, J. Netw. Comput. Appl. 115 (2018) 70–85.

[33] Zhou, et al., Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions, ACM Comput. Surv. 51 (1) (2018) 1–38.

[34] Aliyu, et al., Mobile cloud computing: taxonomy and challenges, J. Comput. Netw. Commun. 2020 (2020) 1–23.

[35] S. others, Computational intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues, J. Inf. Secur. Appl. 55 (2020) 102582.

[36] Rahmani, et al., Towards data and computation offloading in mobile cloud computing: taxonomy, overview, and future directions, Wirel. Pers. Commun. 119 (2021) 147–185.

[37] Maray, et al., Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues, Mob. Inf. Syst. 2022 (2022).

[38] K. Parshotam, Crowd computing: a literature review and definition, in: Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, 2013, pp. 121–130.

[39] Murray, et al., The case for crowd computing, in: Proceedings of the Second ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds, 2010, pp. 39–44.

[40] Miller, et al., Heads in the cloud, XRDS: Crossroads, ACM Mag. Stud. 17 (2) (2010) 27–31.

[41] Schneider, et al., CSCWD: Five characters in search of crowds, in: Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design, CSCWD, IEEE, 2012, pp. 634–641.

[42] Muhammadi, et al., Crowd computing: a survey, 2013, arXiv preprint arXiv:1301.2774.

[43] Rahmany, et al., A review of desktop grid computing middlewares on non-dedicated resources, J. Theor. Appl. Inf. Technol. 98 (10) (2020) 1654–1663.

[44] Choi, et al., Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment, in: Third IEEE International Symposium on Network Computing and Applications, 2004.(NCA 2004). Proceedings, IEEE, 2004, pp. 366–371.

[45] Chien, et al., Entropia: architecture and performance of an enterprise desktop grid system, J. Parallel Distrib. Comput. 63 (5) (2003) 597–610.

[46] Posypkin, et al., Using BOINC desktop grid to solve large scale SAT problems, Comput. Sci. 13 (1) (2012) 25.

[47] M.J. Litzkow, Remote Unix: Turning idle workstations into cycle servers, in: Proceedings of the Summer USENIX Conference, 1987, pp. 381–384.

[48] Sarmenta, et al., Bayanihan: Building and studying web-based volunteer computing systems using Java, Future Gener. Comput. Syst. 15 (5–6) (1999) 675–686.

[49] D.P. Anderson, Boinc: A system for public-resource computing and storage, in: Fifth IEEE/ACM International Workshop on Grid Computing, IEEE, 2004, pp. 4–10.

[50] Ivashko, et al., A survey of desktop grid scheduling, IEEE Trans. Parallel Distrib. Syst. 29 (12) (2018) 2882–2895.

[51] Guo, et al., From participatory sensing to mobile crowd sensing, in: 2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PERCOM WORKSHOPS, IEEE, 2014, pp. 593–598.

[52] Liu, et al., Data-oriented mobile crowdsensing: A comprehensive survey, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2849–2885.

[53] Rana, et al., Ear-phone: an end-to-end participatory urban noise mapping system, in: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, 2010, pp. 105–116.

[54] Ganti, et al., Mobile crowdsensing: current state and future challenges, IEEE Commun. Mag. 49 (11) (2011) 32–39.

[55] Yang, et al., Security and privacy in mobile crowdsourcing networks: challenges and opportunities, IEEE Commun. Mag. 53 (8) (2015) 75–81.

[56] Wang, et al., Mobile crowdsourcing: framework, challenges, and solutions, Concurr. Comput.: Pract. Exp. 29 (3) (2017) e3789.

[57] Malo-Perisé, et al., The "socialized architecture": A software engineering approach for a new cloud, Sustainability 14 (4) (2022) 2020.

[58] N. Kratzke, Volunteer down: How covid-19 created the largest idling supercomputer on earth, Future Internet 12 (6) (2020) 98.

[59] Tapparello, et al., Volunteer computing on mobile devices: State of the art and future research directions, in: Mobile Computing and Wireless Networks: Concepts, Methodologies, Tools, and Applications, IGI Global, 2016, pp. 2171–2198.

[60] Toczé, et al., A taxonomy for management and optimization of multiple resources in edge computing, Wirel. Commun. Mob. Comput. 2018 (2018).

[61] Dillon, et al., Cloud computing: issues and challenges, in: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Ieee, 2010, pp. 27–33.

[62] Buyya, et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.
[63] Hong, et al., Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms, ACM Comput. Surv. 52 (5) (2019) 1–37.
[64] Khan, et al., Edge computing: A survey, Future Gener. Comput. Syst. 97 (2019) 219–235.
[65] Cornelius, et al., Anonysense: privacy-aware people-centric sensing, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, 2008, pp. 211–224.
[66] Cuervo, et al., CrowdLab: An architecture for volunteer mobile testbeds, in: 2011 Third International Conference on Communication Systems and Networks, COMSNETS 2011, IEEE, 2011, pp. 1–10.
[67] Schildt, et al., Candis: Heterogenous mobile cloud framework and energy cost-aware scheduling, in: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, IEEE, 2013, pp. 1986–1991.
[68] Datta, et al., ANGELS: A framework for mobile grids, in: 2014 Applications and Innovations in Mobile Computing, AIMoC, IEEE, 2014, pp. 15–20.
[69] Xu, et al., Ocelot: A wireless sensor network and computing engine with commodity palmtop computers, in: 2013 International Green Computing Conference Proceedings, IEEE, 2013, pp. 1–8.
[70] Parmar, et al., JUniGrid: A simplistic framework for integration of mobile devices in heterogeneous grid computing, Int. J. Multidiscip. Sci. Eng. 4 (1) (2013) 10–15.
[71] Das, et al., PRISM: platform for remote sensing using smartphones, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, 2010, pp. 63–76.
[72] Agapie, et al., Seeing Our Signals: Combining location traces and web-based models for personal discovery, in: Proceedings of the 9th Workshop on Mobile Computing Systems and Applications, 2008, pp. 6–10.
[73] Deng, et al., Livecompare: grocery bargain hunting through participatory sensing, in: Proceedings of the 10th Workshop on Mobile Computing Systems and Applications, 2009, pp. 1–6.
[74] Gaonkar, et al., Micro-blog: sharing and querying content through mobile phones and social participation, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, 2008, pp. 174–186.
[75] Lu, et al., Bubble-sensing: Binding sensing tasks to the physical world, Pervasive Mob. Comput. 6 (1) (2010) 58–71.
[76] Gordienko, et al., Synergy of volunteer measurements and volunteer computing for effective data collecting, processing, simulating and analyzing on a worldwide scale, in: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, IEEE, 2015, pp. 193–198.
[77] Ba, et al., Mobile computing-A green computing resource, in: 2013 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2013, pp. 4451–4456.
[78] Arslan, et al., Computing while charging: Building a distributed computing infrastructure using smartphones, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, 2012, pp. 193–204.
[79] Shi, et al., Serendipity: Enabling remote computing among intermittently connected mobile devices, in: Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2012, pp. 145–154.
[80] Habak, et al., Femto clouds: Leveraging mobile devices to provide cloud service at the edge, in: 2015 IEEE 8th International Conference on Cloud Computing, IEEE, 2015, pp. 9–16.
[81] Mtibaa, et al., Towards resource sharing in mobile device clouds: Power balancing across mobile devices, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 51–56.
[82] Al Noor, et al., Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives, in: 2014 IEEE 7th International Conference on Cloud Computing, IEEE, 2014, pp. 200–207.
[83] E.E. Marinelli, Hyrax: Cloud Computing on Mobile Devices Using Mapreduce, Tech. Rep., Carnegie-mellon univ Pittsburgh PA school of computer science, 2009.
[84] Jassal, et al., Unity: Collaborative downloading content using co-located socially connected peers, in: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops, IEEE, 2013, pp. 66–71.
[85] Chu, et al., Mobile ogsi. net: Grid computing on mobile devices, in: Fifth IEEE/ACM International Workshop on Grid Computing, IEEE, 2004, pp. 182–191.
[86] Fernando, et al., Honeybee: A programming framework for mobile crowd computing, in: Mobile and Ubiquitous Systems: Computing, Networking, and Services: 9th International Conference, MobiQuitous 2012, Beijing, China, December 12-14, 2012. Revised Selected Papers 9, Springer, 2013, pp. 224–236.
[87] Agarwal, et al., DRAP: A decentralized public resourced cloudlet for Ad-hoc networks, in: 2015 IEEE 4th International Conference on Cloud Networking, CloudNet, IEEE, 2015, pp. 309–314.
[88] Li, et al., A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing, IEEE Trans. Ind. Inform. 15 (7) (2019) 4225–4234.
[89] Liu, et al., Dependency-aware task scheduling in vehicular edge computing, IEEE Internet Things J. 7 (6) (2020) 4961–4971.
[90] Ni, et al., Resource allocation strategy in fog computing based on priced timed petri nets, Ieee Internet Things J. 4 (5) (2017) 1216–1228.
[91] Wadhwa, et al., TRAM: Technique for resource allocation and management in fog computing environment, J. Supercomput. 78 (1) (2022) 667–690.
[92] Agarwal, et al., An efficient architecture and algorithm for resource provisioning in fog computing, Int. J. Inf. Eng. Electron. Bus. 8 (1) (2016) 48.
[93] Wu, et al., An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing, Future Gener. Comput. Syst. 117 (2021) 498–509.
[94] Liu, et al., A framework of fog computing: Architecture, challenges, and optimization, IEEE Access 5 (2017) 25445–25454.
[95] Liu, et al., A task scheduling algorithm based on classification mining in fog computing environment, Wirel. Commun. Mob. Comput. 2018 (2018).
[96] Bian, et al., Online task scheduling for fog computing with multi-resource fairness, in: 2019 IEEE 90th Vehicular Technology Conference, VTC2019-Fall, IEEE, 2019, pp. 1–5.
[97] Zeng, et al., Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (6) (2020) 3247–3257.
[98] Liu, et al., Joint optimization of path planning and resource allocation in mobile edge computing, IEEE Trans. Mob. Comput. 19 (9) (2019) 2129–2144.
[99] Ren, et al., Collaborative cloud and edge computing for latency minimization, IEEE Trans. Veh. Technol. 68 (5) (2019) 5031–5044.
[100] Qiu, et al., Avr: Augmented vehicular reality, in: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, 2018, pp. 81–95.
[101] Zhao, et al., Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing, in: 2017 IEEE International Conference on Communications, ICC, IEEE, 2017, pp. 1–7.
[102] Liu, et al., Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, IEEE Trans. Wireless Commun. 18 (1) (2018) 695–708.
[103] Kuang, et al., Partial offloading scheduling and power allocation for mobile edge computing systems, IEEE Internet Things J. 6 (4) (2019) 6774–6785.
[104] Xing, et al., Joint task assignment and resource allocation for D2D-enabled mobile-edge computing, IEEE Trans. Commun. 67 (6) (2019) 4193–4207.
[105] Mao, et al., Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in: 2017 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2017, pp. 1–6.
[106] Saleem, et al., Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing, IEEE Trans. Wireless Commun. 20 (1) (2020) 360–374.
[107] Yuan, et al., Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems, IEEE Trans. Autom. Sci. Eng. 18 (3) (2020) 1277–1287.
[108] Alameddine, et al., Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing, IEEE J. Sel. Areas Commun. 37 (3) (2019) 668–682.
[109] Nath, et al., Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems, Intell. Converged Netw. 1 (2) (2020) 181–198.
[110] Li, et al., Radio and computing resource allocation with energy harvesting devices in mobile edge computing environment, Comput. Commun. 145 (2019) 193–202.
[111] Xu, et al., Dynamic task scheduling algorithm with deadline constraint in heterogeneous volunteer computing platforms, Future Internet 11 (6) (2019) 121.
[112] Hoseiny, et al., Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system, ACM Trans. Internet Technol. (TOIT) 21 (4) (2021) 1–21.
[113] Panadero, et al., A simheuristic approach for resource allocation in volunteer computing, in: 2017 Winter Simulation Conference, WSC, IEEE, 2017, pp. 1479–1490.
[114] Panadero, et al., Multi criteria biased randomized method for resource allocation in distributed systems: Application in a volunteer computing system, Future Gener. Comput. Syst. 82 (2018) 29–40.
[115] Pham, et al., Joint node selection and resource allocation for task offloading in scalable vehicle-assisted multi-access edge computing, Symmetry 11 (1) (2019) 58.
[116] Rubab, et al., Bin packing multi-constraints job scheduling heuristic for heterogeneous volunteer grid resources, in: The Fourth International Conference on Computer Science & Computational Mathematics, ICCSCM 2015, 2015.
[117] Ali, et al., A volunteer-supported fog computing environment for delay-sensitive iot applications, IEEE Internet Things J. 8 (5) (2020) 3822–3830.
[118] Feng, et al., Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems, IEEE Trans. Wireless Commun. 19 (6) (2020) 4321–4334.

[119] Guo, et al., Energy-efficient resource allocation for multi-user mobile edge computing, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.
[120] Hu, et al., Dynamic request scheduling optimization in mobile edge computing for IoT applications, IEEE Internet Things J. 7 (2) (2019) 1426–1437.
[121] Zhang, et al., Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing, IEEE Internet Things J. 6 (2) (2018) 3688–3699.
[122] Yu, et al., Joint subcarrier and CPU time allocation for mobile edge computing, in: 2016 IEEE Global Communications Conference, GLOBECOM, IEEE, 2016, pp. 1–6.
[123] Zhang, et al., Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN, IEEE Internet Things J. 7 (4) (2020) 3282–3299.
[124] Zhang, et al., Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing, IEEE Access 6 (2018) 19324–19337.
[125] Samanta, et al., Dyme: Dynamic microservice scheduling in edge computing enabled IoT, IEEE Internet Things J. 7 (7) (2020) 6164–6174.
[126] Wang, et al., Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing, IEEE Trans. Cybern. 50 (9) (2019) 3984–3997.
[127] Hu, et al., UAV-assisted relaying and edge computing: Scheduling and trajectory optimization, IEEE Trans. Wireless Commun. 18 (10) (2019) 4738–4752.
[128] Wang, et al., Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing, Sustain. Comput.: Inform. Syst. 21 (2019) 154–164.
[129] Zhang, et al., Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing, IEEE Trans. Veh. Technol. 69 (2) (2019) 2114–2125.
[130] Liu, et al., Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing, IEEE Trans. Commun. 67 (6) (2019) 4132–4150.
[131] Yu, et al., Energy-efficient task offloading and resource scheduling for mobile edge computing, in: 2018 IEEE International Conference on Networking, Architecture and Storage, NAS, IEEE, 2018, pp. 1–4.
[132] Li, et al., Optimizing resources allocation for fog computing-based internet of things networks, IEEE Access 7 (2019) 64907–64922.
[133] Yi, et al., Joint resource allocation for device-to-device communication assisted fog computing, IEEE Trans. Mob. Comput. 20 (3) (2019) 1076–1091.
[134] Lei, et al., Multiuser resource control with deep reinforcement learning in IoT edge computing, IEEE Internet Things J. 6 (6) (2019) 10119–10133.
[135] Guo, et al., Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems, IEEE Access 7 (2019) 78685–78697.
[136] Liu, et al., Resource allocation with edge computing in IoT networks via machine learning, IEEE Internet Things J. 7 (4) (2020) 3415–3426.
[137] Luo, et al., HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning, IEEE Trans. Wireless Commun. 19 (10) (2020) 6535–6548.
[138] Yin, et al., Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing, IEEE Trans. Ind. Inform. 14 (10) (2018) 4712–4721.
[139] Mukherjee, et al., Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency, IEEE Commun. Lett. 24 (2) (2019) 307–311.
[140] Wang, et al., A dynamic resource scheduling scheme in edge computing satellite networks, Mob. Netw. Appl. 26 (2) (2021) 597–608.
[141] Xiong, et al., Resource allocation based on deep reinforcement learning in IoT edge computing, IEEE J. Sel. Areas Commun. 38 (6) (2020) 1133–1146.
[142] Li, et al., Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment, Future Gener. Comput. Syst. 95 (2019) 249–264.
[143] Daoud, et al., TACRM: trust access control and resource management mechanism in fog computing, Hum.-Cent. Comput. Inf. Sci. 9 (1) (2019) 1–18.
[144] Huang, et al., Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing, Peer-to-Peer Netw. Appl. 13 (5) (2020) 1776–1787.
[145] Choudhari, et al., Prioritized task scheduling in fog computing, in: Proceedings of the ACMSE 2018 Conference, 2018, pp. 1–8.
[146] Mutlag, et al., MAFC: Multi-agent fog computing model for healthcare critical tasks management, Sensors 20 (7) (2020) 1853.
[147] Feng, et al., Dynamic network slicing and resource allocation in mobile edge computing systems, IEEE Trans. Veh. Technol. 69 (7) (2020) 7863–7878.
[148] Meng, et al., Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 2287–2295.
[149] Sun, et al., Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II, Wirel. Pers. Commun. 102 (2) (2018) 1369–1385.
[150] Zeng, et al., Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system, IEEE Trans. Comput. 65 (12) (2016) 3702–3712.
[151] Tang, et al., Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud–edge environment, Comput. Commun. 134 (2019) 70–82.
[152] Yang, et al., Deep reinforcement learning based resource allocation in low latency edge computing networks, in: 2018 15th International Symposium on Wireless Communication Systems, ISWCS, IEEE, 2018, pp. 1–5.
[153] Sun, et al., Joint communication and computing resource allocation in vehicular edge computing, Int. J. Distrib. Sens. Netw. 15 (3) (2019) 1550147719837859.
[154] Wang, et al., Coupling resource management based on fog computing in smart city systems, J. Netw. Comput. Appl. 135 (2019) 11–19.
[155] Rafique, et al., A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing, IEEE Access 7 (2019) 115760–115773.

Peizhe Ma received the B.ICT. degree in software development in 2022, and currently in her final semester of B. ICT. (Hons.) degree with a focus on resource allocation challenges in the edge computing and volunteer computing to meet the time requirement of time sensitive Internet of Things.

Saurabh Garg received a Ph.D. degree from The University of Melbourne. He is currently a Lecturer at the University of Tasmania, Australia. He has authored over 40 papers in highly cited journals and conferences. His research interests include resource management, scheduling, utility and grid computing, cloud computing, green computing, wireless networks, and ad hoc networks. He received various special scholarships for his Ph.D. candidature.

Mutaz Barika has a Ph.D. in Information Technology from the University of Tasmania, Australia. He is currently a Lecturer and Unit Coordinator at Crown Institute of Higher Education, Australia. He holds Microsoft Certified Solutions Expert: Cloud Platform and Infrastructure Charter Member Certificate. He has published more than 20 scientific papers in international conferences and journals, and looks forward to developing collaborative projects in the IoT-cloud continuum. His current research interests include Big Data, Big Data Workflow, IoT, Cloud Computing and Data Security.