2022 IEEE International Conference on Edge Computing and Communications (EDGE) | 978-1-6654-8140-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/EDGE55608.2022.00031

Physics-Inspired Mobile Cloudlet Placement in Next-Generation Edge Networks

Dixit Bhatta
Department of Computer and Information Sciences
University of Delaware
Newark, DE, USA
dixit@udel.edu

Lena Mashayekhy
Department of Computer and Information Sciences
University of Delaware
Newark, DE, USA
mlena@udel.edu

***Abstract*—IoT (Internet-of-Things) devices require both reliable, ultra-low latency connection and on-demand access to computing resources in their vicinity. Edge computing can provide nearby computing resources through offloading to cloudlets. However, challenges lie in providing stable services in big cities, emergency situations, and places lacking sufficient infrastructure due to dynamic IoT demands and their uncertain mobility, coupled with intermittent or limited connectivity to nearby computing resources. There has been an increasing adoption of UAV-mounted mobile cloudlets to address these issues, where mobile cloudlets can be seamlessly integrated into the existing network infrastructure for offloading and enhanced connectivity. However, adaptively placing mobile cloudlets under continuously evolving conditions is still an open problem. In this paper, we study mobile cloudlet placement in highly dynamic, next-generation edge networks. Our objective is to maintain stable ultra-low-latency services and enhance coverage by optimizing the placement of mobile cloudlets in highly dynamic scenarios. We propose a novel approach for this problem using inspirations from concepts in physics, namely the center of gravity and force of attraction. Our approach emulates how physical bodies maintain balance and adjust their positions when acted upon by dynamic external forces for efficient placement of mobile cloudlets in response to dynamic and uncertain device mobility. The results show that our proposed approach provides high device coverage, excellent energy efficiency, and stable edge services in real-time.**

***Index Terms*—physics-inspired approach; dynamic placements; mobile cloudlets; edge computing; next-generation networks**

I. Introduction

Exciting developments in wireless networks and telecommunications have dramatically impacted many different fields. Nonetheless, continuous efforts to further progress and envision future technologies based on increasing demand for data, flexible computing resources, and connectivity are being made. It is now clear that there will be many novel, cutting-edge, and large-scale applications on billions of connected devices and IoT (Internet-of-Things) endpoints that will be pushing boundaries of the existing infrastructure [1], [2]. For instance, new applications like Extended Reality (XR) have a unique combination of requirements including on-demand services with extremely high network speed and low latency [3]. Edge computing can partially address these requirements by allowing users to consume the computing resources in their vicinity via offloading, thereby reducing the access latency and data traffic to the core infrastructure [4]. The convergence of edge computing with next-generation networks (5G and beyond) can complete the requirements puzzle by providing extremely high network speeds [5], thereby creating the notion of next-generation edge networks.

Nonetheless, covering highly mobile users while providing stable low-latency services as they move across the service region is a complex problem. Locations with limited or disrupted connectivity during natural disasters or generally poor infrastructure cannot sufficiently serve mobile users. Even infrastructures with wide coverage may not meet offloading and ultra-low latency requirements. Next-generation wireless networks with multi-gigabit data rates can still have technical challenges such as signal reflection, path loss, and blockages [5]. On the other hand, offloading to static edge servers (or cloudlets) is equally challenging and leads to frequent service and task migrations for mobile users. Accordingly, the long-term, static placement of cloudlets does not meet all the requirements of mobile users in next-generation edge networks. Thus, we can adequately serve highly mobile users with dynamic demands only when efficient approaches to dynamically place the cloudlets are available.

Mobile cloudlets mounted on Unmanned Aerial Vehicles (UAVs) have been well-studied and already under adoption in the industry [6], [7] to enhance user experience and enable new innovative applications. Federal Aviation Authority (FAA) and Verizon's Skyward currently collaborate to test cellular-connected UAVs [8]. Some studies have proposed UAVs to complement wireless backhaul networks [9] and even an entire cellular infrastructure based on a hierarchical deployment of UAVs [10]. UAV-mounted cloudlets can be flexibly deployed as the user demands and distributions change over a region. This is especially applicable to support compute-intensive latency-critical computations in disaster scenarios [7], massive demand areas such as stadiums, and emerging applications including drone air-delivery. For instance, dispersed teams of rescue workers can use XR headgears in inaccessible or partially accessible landscapes for path finding during natural disasters, and cognitive assistance such as identifying objects, people, and threats. These rapidly evolving compute-intensive applications (object detection, risk assessment, and navigation) need ultra-low latency and stable offloading services so that well-informed decisions can be made in a matter of milliseconds, which are otherwise not possible under damaged or distant infrastructure, and where emergency support vehicles and equipment cannot reach. Mobile cloudlets meet these critical requirements by directly bringing resources to the rescuers in response to their uncertain movements.

The purpose of this paper is to present an ingenious placement of mobile cloudlets in next-generation edge networks to provide ultra-low latency services for the emerging applications considering realistic constraints. We aim to provide stable services to the users through mobile cloudlets when user distribution and demands change rapidly.

We design a novel approach, called Physics-inspired Mobile Cloudlet Placement approach, PMCP, to address these issues by borrowing the concepts of "center of gravity" and "force of attraction" from physics. Our proposed approach responds to uncertain device mobility in real-time. PMCP not only removes the overhead of predicting device locations but also does not require the acquisition of sensitive mobility data to make those predictions.

As a part of our study, we define an optimal integer programming formulation of the mobile cloudlet placement problem, OMCP, and implement the classic DBSCAN algorithm [11] with path planning to serve as our two performance benchmarks. We perform extensive experiments by creating multiple scenarios, including both 2D and 3D user distribution and mobility, based on real mobility traces obtained from KTH Walkers dataset [12]. We then compare PMCP's results with OMCP and DBSCAN. The results show that our approach achieves high coverage values, superior energy efficiency, and stable services for all scenarios in real-time.

The rest of the paper is organized as follows. In Section II, we discuss the state-of-the-art research in this domain. In Section III, we introduce the mobile cloudlet placement problem and formulate a mathematical optimization model. In Section IV, we present our proposed approach, PMCP, in detail. In Section V, we evaluate the performance of PMCP by extensive experiments. In Section VI, we summarize our results and present possible directions for future research.

II. Related Work

Numerous studies have been conducted to address the cloudlet placement problem in a static scenario. Clustering-based approaches have been proposed by Kang *et al.* [13] and Jia *et al.* [14]. Greedy approaches include the likes of Zeng *et al.* [15] and Yao *et al.* [16]. Other static approaches include a search-based algorithm by Wang *et al.* [17], an energy-aware heuristic approach by Li *et al.* [18], and a coverage maximization approach [19]. Lu *et al.* [20] studied the robust placement of edge servers under failure scenarios, and Fan and Ansari [21] proposed another heuristic algorithm for cost-aware placement to obtain sub-optimal solutions. Many of these studies lack heterogeneity. Bhatta and Mashayekhy proposed a meta-heuristic approach [22] and a bifactor approximation algorithm [23] for heterogeneous cloudlet placement. However, these studies do not consider the mobility of the users, the cloudlets, or both. These approaches are primarily designed for the permanent placement of cloudlets, hence, they are not suitable for scenarios where placements need to be updated in very short time intervals.

There are related studies on UAV positioning and mobility models in flying ad-hoc networks (FANET) and node placement in wireless sensor networks (WSN) [24], [25]. Studies on WSN node placement focus on maximizing the surveillance area or geographical coverage of the nodes [24]. However, our problem is concerned with covering actual devices that need computing resources and connectivity to services. Moreover, higher geographic coverage may not even lead to better device coverage in a dynamic environment where multiple sub-regions may not even have any users. Likewise, FANET positioning and mobility models are essentially geographical deployments, and they too do not follow the users or their demands. Many models are in fact based on time-based, topological, and even random positioning of UAVs, agnostic of user mobility or demand [25]. Hence, existing studies in both of these domains insufficiently address our problem.

When it comes to mobile cloudlet placements, there are extremely limited studies on dynamic or online placement of cloudlets since the primary infrastructure has been most often perceived to be static and perennial. However, mobile cloudlets have become increasingly relevant due to ubiquitous mobile applications today, rendering a necessity for more studies. Xiang *et al.* [26] proposed an adaptive cloudlet placement approach for mobile applications. Their approach identifies gathering regions of mobile devices using position clustering and generates mobility paths of the cloudlets to new locations based on the shortest distance. Their approach inherently does not capture the mobility of each cloudlet as a part of the main decision, which is to determine the cluster centers instead. They recalculate cluster centers in each time slot and only calculate the mobility paths after the new cluster centers are established. Moreover, their approach looks at the placement from a 2D perspective with homogeneous cloudlets and is computationally heavy, limiting its applicability significantly, especially for real-time scenarios. Zhang *et al.* [27] presented another adaptive cloudlet placement approach that directly improves on the previous study. They proposed a covering-based clustering technique to determine cloudlet placement locations. They also made the covering algorithm parallel on Spark to speed it up. Nonetheless, the limitations stay the same since their core assumptions are exactly the same as the previous study. Jin *et al.* [28] proposed another clustering-based approach that deploys cloudlets dynamically based on the geographic location and the number of tasks generated by devices. However, their approach again relies on establishing new device cluster centers in each time window and then moving the cloudlets to their closest destination centers.

Wang *et al.* [29] proposed an online algorithm that dispatches UAV-mounted edge servers by identifying UAV hover locations to complement existing infrastructure during heavy usage by maximizing the number of served tasks. However, they do not consider separating distance between the UAVs and possible interference when they are placed too close to each other, around the same location. Although the UAVs hover at different heights, all devices are assumed to be on the ground. Hence, the coverage is observed in terms of the 2D influence radius on the ground. Moreover, they do not consider the concept of service stability. A dynamic approach for mobile environments by Yuan *et al.* [30] uses deep learning for virtual edge node placements in edge cloud systems. Their combination of deep-learning-based predictions and hierarchical-clustering-based placement approach is catered more towards dynamic service placement than the actual placement of mobile cloudlets. Service or application placement approaches have inherently different properties which are not suitable for our problem. Moreover, user mobility is highly uncertain, and even deep learning may lead to inaccuracy and consequently high latency services. Furthermore, considering the importance of user privacy, our approach does not rely on learning or predicting user mobility patterns.

In summary, apart from [29], none of the existing dynamic placement approaches consider cloudlet capacities or user to cloudlet assignments as a part of their problem formulation. Most importantly, in all existing studies, the movement of the individual cloudlets is not directly guided by changes in device positions and demands in their immediate surrounding. In our approach, the mobile cloudlets are aware of and respond to changes around them in real-time. There is no prediction involved, which reduces the run-time overhead. Similarly, sensitive long-term mobility data required for those predictions do not need to be acquired or processed by our approach.

III. Mobile Cloudlet Placement Problem

This paper aims to continuously place and readjust the locations of mobile cloudlets in a region to maximize device coverage and maintain stable services, given the uncertain changes in user demands and movements. As such, we model the region as a finite 3D grid within which the devices and the mobile cloudlets move. The 3D grid is made up of smaller contiguous 3D sub-divisions called cells.

The 3D grid is a collection of non-overlapping equal-sized cells or cubes denoted by $\mathcal{C} = \{c_1, c_2, \ldots, c_k, \ldots, c_n\}$. Note that the geometric representation of the cells can be other 3D shapes such as a hexagonal prism or a sphere. For simplicity, we consider cubes. Each cell represents the 3D coverage of a unit cloudlet within which ultra-low latency can be offered. The interconnected cells make the contiguous

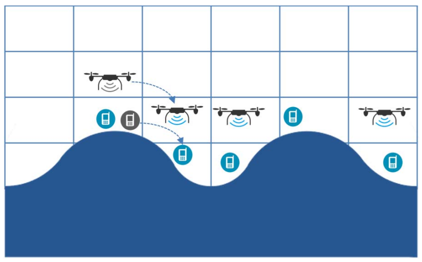


Fig. 1: Mobile cloudlet placement scenario in 2D

3D region. The cells, represented using 3D coordinate axes, are always indexed from top-left to bottom-right with the index denoting the area they cover. The mobile cloudlets and devices may change their locations from cell to cell over time $\tau = \{1, 2, \ldots, t, \ldots, T\}$. The cloudlets move between the cells to follow the changes in the device distribution, as shown in Figure 1 (shown in 2D for simplicity).

We denote the set of heterogeneous mobile cloudlets by $\mathcal{M} = \{m_1, m_2, \ldots, m_j, \ldots, m_u\}$. Each mobile cloudlet $m_j \in \mathcal{M}$ is represented by a 3-tuple $m_j = \{\mu_j^t, r_j, \Lambda_j\}$ denoting its attributes: μ_j^t is the service capacity (in resource units) at time t, r_j is the 3D coverage (3D cell dimensions), and Λ_j is the maximum cell distance the cloudlet can travel within a time slot t. Note that the resources indicated by service capacity μ_j^t can be processing, memory, or storage individually or as a combination. We assume a single resource for simplicity and without loss of generality. Similarly, the maximum cell travel Λ_j depends on multiple parameters such as the speed of the mobile cloudlet, power consumption/available battery, and the weight carried by the mobile cloudlet. This is used to set a constraint on general mobility of the cloudlets so that they follow a realistic travel path and travel a limited distance in every time slot while avoiding collisions during placement.

Likewise, we denote the set of heterogeneous devices by $\mathcal{D} = \{d_1, d_2, \ldots, d_i, \ldots, d_v\}$. Each device $d_i \in \mathcal{D}$ is represented by a 3-tuple $d_i = \{\delta_i^t, c_i^t, \lambda_i^t\}$ denoting its attributes: δ_i^t is the demand (in resource units) at time t, c_i^t is the 3D cell of the mobile device at time t, and λ_i^t is an ordered triple denoting the 3D coordinates of the mobile device at time t.

For simplicity of modeling, if a cloudlet fails at any instant, the cloudlet capacity becomes 0. If it has recovered, its capacity comes back to non-zero. The same applies to a device when it is disconnected or connected, and the device demand goes to 0 or non-zero, respectively. These can also be treated as a device going out of bounds and returning back to the service region. We assume that these values are set by the underlying sensor networks. In addition, all connected devices are assumed to receive ultra-low latency services if they are within the 3D coverage regardless of their absolute distance from the mobile cloudlet.

Our goal is to maximize the coverage of the devices in the region (number of connected devices) with ultra-low latency services by moving the cloudlets in response to the device mobility. We define the following decision variables:

$$\alpha_{jk}^t = \begin{cases} 1 & \text{if cloudlet } m_j \text{ is assigned to cell } c_k \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

$$\gamma_{jkl}^t = \begin{cases} 1 & \text{if cloudlet } m_j \text{ moves from } c_k \text{ to } c_l \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

$$\beta_{ij}^t = \begin{cases} 1 & \text{if device } d_i \text{ is assigned to cloudlet } m_j \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

We mathematically formulate the Optimal Mobile Cloudlet Placement problem (OMCP) as an Integer Program (IP) as follows:

$$\max \sum_{t=1}^{T} \sum_{i=1}^{v} \sum_{j=1}^{u} \beta_{ij}^{t} \tag{1}$$

Subject to:

$$\sum_{j=1}^{u} \sum_{k=1}^{n} \alpha_{jk}^{t} = |\mathcal{M}|, \quad \forall t \in \tau \tag{2}$$

$$\sum_{k=1}^{n} \Delta(c_i^t, c_k)(\beta_{ij}^t + \alpha_{jk}^t - 1) \le r_j, \forall d_i \in \mathcal{D}, m_j \in \mathcal{M}, t \in \tau \tag{3}$$

$$\sum_{i=1}^{v} \delta_i^t \beta_{ij}^t \le \mu_j^t, \quad \forall m_j \in \mathcal{M}, t \in \tau \tag{4}$$

$$\gamma_{jkl}^t \ge \alpha_{jk}^{t-1} + \alpha_{jl}^t - 1, \forall m_j \in \mathcal{M}, c_k, c_l \in \mathcal{C}, t \in \{2, \ldots, T\} \tag{5}$$

$$\gamma_{jkl}^t \le \alpha_{jk}^{t-1}, \quad \forall m_j \in \mathcal{M}, c_k, c_l \in \mathcal{C}, t \in \{2, \ldots, T\} \tag{6}$$

$$\gamma_{jkl}^t \le \alpha_{jl}^t, \quad \forall m_j \in \mathcal{M}, c_k, c_l \in \mathcal{C}, t \in \{2, \ldots, T\} \tag{7}$$

$$\Delta(c_k, c_l)\gamma_{jkl}^t \le \Lambda_j, \forall c_k, c_l \in \mathcal{C}, m_j \in \mathcal{M}, t \in \{2, \ldots, T\} \tag{8}$$

$$\sum_{j=1}^{u} \alpha_{jk}^t \le 1, \quad \forall c_k \in \mathcal{C}, t \in \tau \tag{9}$$

$$\sum_{k=1}^{n} \alpha_{jk}^t \le 1, \quad \forall m_j \in \mathcal{M}, t \in \tau \tag{10}$$

$$\beta_{ij}^t \le \sum_{k=1}^{n} \alpha_{jk}^t, \quad \forall d_i \in \mathcal{D}, m_j \in \mathcal{M}, t \in \tau \tag{11}$$

$$\sum_{j=1}^{u} \beta_{ij}^t \le 1, \quad \forall d_i \in \mathcal{D}, t \in \tau \tag{12}$$

$$\gamma_{jkl}^t \in \{0, 1\}, \quad \forall m_j \in \mathcal{M}, c_k, c_l \in \mathcal{C}, t \in \tau \tag{13}$$

$$\alpha_{jk}^t, \beta_{ij}^t \in \{0, 1\}, \quad \forall m_j \in \mathcal{M}, d_i \in \mathcal{D}, c_k \in \mathcal{C}, t \in \tau \tag{14}$$

The objective function shown in Eq (1) maximizes the coverage of the devices, i.e., mapping of the devices to the placed cloudlets. Constraints (2) ensure that the total number of cloudlets placed in the region is equal to the number of available cloudlets. Constraints (3) guarantee that each covered device must be within the coverage radius of some cloudlet. The constraint indicates that a device d_i currently at cell c_i^t if mapped to a cloudlet m_j placed at a cell c_k, the distance between their cells must be less than or equal to the 3D coverage r_j of the cloudlet. Constraints (4) satisfy supply and demand based on each cloudlet capacity and its covered device demands. The sum of all device demands assigned to a cloudlet must be less than or equal to the resources of that cloudlet. Constraints (5) ensure that a cloudlet moves from one cell to another at t only if its cell association changes over time $t-1$ and t. Constraints (6) and (7) ensure that a cloudlet moves from one cell to another at t only if it was placed in the source cell at $t-1$ and in the destination cell at t. Constraints (8) ensure that a cloudlet moves no further than the maximum distance it can travel within time slot t. Constraints (9) ensure that at most one cloudlet is placed at any cell in the region. Constraints (10) ensure that a cloudlet can only be placed at a single cell. Constraints (11) guarantee that a device can only be served by a cloudlet that is placed in the grid. Constraints (12) guarantee that each device is served by at most one cloudlet. Finally, constraints (13) and (14) ensure the integrality requirements of the decision variables.

IV. Physics-Inspired Mobile Cloudlet Placement

We propose a novel approach to place and dynamically change the locations of the mobile cloudlets in response to dynamic and uncertain device mobility. Our approach, called Physics-Inspired Mobile Cloudlet Placement (PMCP), is inspired by the concept of the *center of gravity* and *force of attraction*. These concepts have been used for weighted facility location [31] and clustering approaches [32], [33], respectively, in the fields such as operations research and machine learning. The latter especially applies to robust adaptive clustering in distributed networks [33]. In edge computing, the idea of attractive and repulsive forces has been used to move data-collecting mobile edge nodes closer to trustworthy nodes and away from untrustworthy nodes in their overall travel path [34]. Although our work is markedly different from these individual studies, all of them suggest that our approach is highly suitable for the dynamic placement of mobile cloudlets under uncertainty.

PMCP performs the initial mobile cloudlet placement based on the center of gravity to evenly balance the location of the cloudlets. Such a balanced initial placement allows cloudlets to be placed around weighted centers of device demands, enhancing potential coverage. Subsequently, PMCP readjusts the locations of the mobile cloudlets in real-time by moving them using the force of attraction to follow the sub-regions with high user concentration. Since cloudlets follow the devices, devices are likely to be served by the same cloudlets over time, leading to less migration and switching services between the cloudlets, which is significantly important in providing low-latency stable edge services. As the cloudlets astutely match the devices' mobility, it can extend the battery life of the cloudlets by avoiding unnecessary cloudlet movements. Next, we explain these parts in detail.

A. Initial Placement

The initial placement by PMCP is inspired by the center of gravity method. The goal of the initial placement is to minimize the weighted distance of the devices from the mobile cloudlets that will be placed in the 3D grid. PMCP estimates the weighted centers of gravity for the devices in the grid based on the demands and the cell locations of the devices at time $\tau = 1$. The estimated centers of gravity are iteratively improved until they converge. The cloudlets are then initially placed at the central cell locations of the converged centers of gravity. The initial placement by PMCP is described in Algorithm 1.

In Algorithm 1, individual demands of the devices are first aggregated per cell and stored in list A. Thereafter, a list of u

Algorithm 1 PMCP - Initialization

```
1: Input: $\mathcal{M}, \mathcal{C}, \mathcal{D}$
2: A ← AggregateDemandsByCell($\mathcal{C}, \mathcal{D}$)
3: CoG ← TopCells(A, $u$)                ▷ Top $u$ cells in A
4: CoGMap ← CoGDeviceMappings($\mathcal{D}$, CoG)
5: CoGMap⋆ = ∅
6: while CoGMap⋆ != CoGMap do
7:     CoG = ∅
8:     for $\{c_k : [d_a, d_b, ..]\} \in$ CoGMap do
9:         Calculate $(\widehat{X}, \widehat{Y}, \widehat{Z})$ based on $[d_a, d_b, ..]$
10:        CoG ← CoG ∪ the cell closest to $(\widehat{X}, \widehat{Y}, \widehat{Z})$
11:    CoGMap⋆ ← CoGMap
12:    CoGMap ← CoGDeviceMappings($\mathcal{D}$, CoG)
13: CoGMap ← SortDescByDemands(CoGMap)
14: MCP ← MapCloudletsCoGs(CoGMap, $\mathcal{M}$)
15: Output: MCP
```

cells with the highest demands, i.e., CoG is generated from A (line 3). Note that CoG is the list of temporary centers of gravity cells, and $u \leq n$ is the number of all mobile cloudlets. Function CoGDeviceMappings($\mathcal{D}$, CoG) (line 4) returns a mapping between the cells identified in CoG and the devices, where each device is assigned to the closest cell in CoG. This mapping CoGMap $= \{c_k : [d_a, d_b, ..]\}$ represents the temporary mapping of the devices to the centers of gravity CoG cells, which converges through calculations in the next steps of Algorithm 1 (lines 5-12) to provide the initial mobile cloudlet placements.

Once the temporary mapping is done, the algorithm initializes CoGMap⋆ as an empty CoG cells-to-devices mapping to keep track of the changes in the mapping between the centers of gravity CoG and the devices. The recalculation of centers of gravity happens iteratively (lines 6-12) as long as the new CoG and mappings differ from the previous round, i.e., CoGMap and CoGMap⋆ are not equal. For every CoG cell c_k in CoGMap and its corresponding list of devices $[d_a, d_b, ..]$, central coordinates $(\widehat{X}, \widehat{Y}, \widehat{Z})$ are calculated (line 9) using Equation 15.

$$\widehat{X} = \frac{\sum \delta_i x_i}{\sum \delta_i}, \widehat{Y} = \frac{\sum \delta_i y_i}{\sum \delta_i}, \widehat{Z} = \frac{\sum \delta_i z_i}{\sum \delta_i} \tag{15}$$

The sums, denoted by $\sum$, in the equations above are over a set of devices that is supposed to be covered by a single mobile cloudlet. Therefore, $(\widehat{X}, \widehat{Y}, \widehat{Z})$ represents a weighted center of gravity, where a mobile cloudlet will be placed. This implies that the mobile cloudlet will be closer to the devices with higher demand, inadvertently prioritizing cells with more demands, enhancing the overall coverage.

Back to Algorithm 1, the cell closest to $(\widehat{X}, \widehat{Y}, \widehat{Z})$ is added to CoG list as a new center of gravity (line 10). Note that CoG is cleared in line 7 to enlist the newly calculated centers of gravity. In line 11, the previous CoGMap becomes CoGMap⋆, and the new CoGMap is calculated by reassigning devices to the closest cells in the updated CoG (line 12). At the end of each iteration, CoGMap represents temporary CoG cells with their corresponding devices. These iterations continue until the updated CoG and their device mappings, i.e., CoGMap do not change any further.

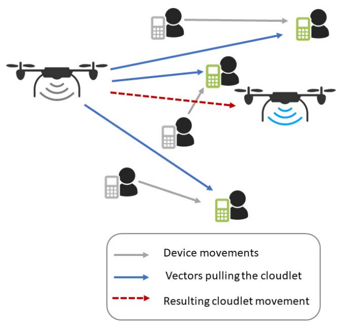


Fig. 2: A mobile cloudlet responding to device mobility

After convergence, CoGMap is then sorted in descending order of the total demand of the devices assigned to each cell in CoG using SortDescByDemands(CoGMap) function (line 13). The mobile cloudlets are mapped one-to-one to the converged CoG based on their size by MapCloudletsCoGs(CoGMap, $\mathcal{M}$) function to provide the mobile cloudlet placements MCP, which is the mapping of mobile cloudlets to the CoG cells $\{(m_a, c_b), \ldots, (m_j, c_k), \ldots\}$ and the output of Algorithm 1.

B. Adaptive Placement

In physics, the force of attraction between two bodies can be formulated in multiple ways based on the type of force. Attractive forces can be the magnetic force, electric force, or the most well-known gravitational force. In our approach, we formulate the force of attraction as a *vector pull force* from the devices acting on the mobile cloudlets which attracts them to their new locations.

Our proposed approach, PMCP, uses this physics concept by considering the devices as moving masses (which need to be served) since they continuously change positions between time slots. In addition, PMCP considers the mobile cloudlets as fixed masses (that are deployed to serve devices) since they are assumed to be at rest in their placed cells until pulled into new cells by the changing device demands and mobility.

Essentially, mobile cloudlets (fixed masses) are initially placed around the devices (moving masses) using the center of gravity method, i.e., Algorithm 1. The mobile cloudlets then move as they are attracted (pulled) by the devices, as illustrated in Figure 2. We use mobile cloudlets' capacities and the mobile devices' demands to represent their physical masses. The mobile cloudlets thus respond to the changes in device locations and demands, and they move to a cell in the direction of the aggregate vector pull forces of the devices. It is important to note that when calculating the forces, mobile cloudlets do not physically interact with each other, neither do the devices.

As we formulate the adaptive part of PMCP, we aim to relocate the mobile cloudlets by estimating their new cell locations at each time slot, denoted by $w_j(t)$ for cloudlet m_j, based on changes in the device specifications at each time slot. Algorithm 2 determines and outputs the new positions of all mobile cloudlets at every time slot based on the overall vector pull

Algorithm 2 PMCP - Adaptive Placement

```
1:  Input: MCP
2:  DCA = DeviceAssignments(MCP)
3:  for $t = 2, 3, \dots, T$ do
4:      for all $m_j \in \mathcal{M}$ do
5:          $\widehat{\mathcal{D}_j}(t)$ = DCA[$m_j$]
6:          Calculate force $\overrightarrow{F_j}(t)$ using Eqn (16)
7:          Assign constrained force $\overrightarrow{F_j}(t)$ to $m_j$ using Eqn (17)
8:          Find estimated location $\widehat{w_j}(t)$ of $m_j$ using Eqn (18)
9:          $w_j(t)$ = ClosestCell($\widehat{w_j}(t)$)
10:     MCP($t$) $\leftarrow w_j(t) \quad \forall m_j \in \mathcal{M}$
11:     DCA = DeviceAssignments(MCP($t$))
12:     Output at $t$: MCP($t$)
```

force acting on them. The force acting on mobile cloudlet m_j is the superposition of all individual forces emerging from each device assigned to it at the end of the previous time slot, obtained by calling DeviceAssignments() function, presented in Algorithm 3. Further details of Algorithm 2 are provided later in this subsection.

As we can observe in Algorithm 3, each device is assigned to its closest feasible mobile cloudlet. The device assignment to the cloudlet is performed as long as the device is within the coverage radius and the mobile cloudlet has enough capacity to meet its demand (lines 1-15). For every device, the algorithm searches for the closest cloudlet that can cover the device and has sufficient remaining capacity to meet its demand (lines 6-12). If such a cloudlet exists, the device is assigned to the cloudlet and the cloudlet capacity is adjusted accordingly (lines 13-15). Otherwise, the devices are simply included in the *uncovered devices* list (line 17). This part of the algorithm provides the actual assignment of the devices which are connected to the mobile cloudlets and are used to calculate the coverage values.

Next, Algorithm 3 performs an extended assignment to assist the cloudlets' mobility (lines 18-22). Here, all devices that could not be covered due to the radius and capacity constraints are assigned to the cloudlets if the devices are within twice the coverage radius of the cloudlet. This is a key feature of the algorithm to make the cloudlets more aware of the devices in their vicinity and move better so that even uncovered devices are covered in later time slots. This also prevents mobile cloudlets with a unit coverage radius from getting stuck in a cell with no device as they will be aware of neighboring cells and can move to serve those uncovered devices in later time slots instead of waiting until some device moves into their coverage radius. The output of Algorithm 3 contains a mapping of all mobile cloudlets to their extended device assignments denoted by DCA.

Algorithm 2 uses the output of Algorithm 3 to obtain a list of devices assigned to cloudlet m_j at the start of time slot t, denoted by $\widehat{\mathcal{D}_j}(t)$ (lines 2-5). Then, Algorithm 2 calculates the total vector pull force acting on mobile cloudlet m_j using the formula given by Equation (16).

$$\overrightarrow{F_j}(t) = \frac{\sum_{\forall d_i(t) \in \widehat{\mathcal{D}_j}(t)} \overrightarrow{\eta}\, \delta_i^t}{\sum_{\forall d_i(t) \in \widehat{\mathcal{D}_j}(t)} \delta_i^t} \tag{16}$$

Algorithm 3 DeviceAssignments()

```
1:  Input: MCP
2:  DCA $\leftarrow \emptyset$, $uncovered\_devices \leftarrow \emptyset$
3:  for $d_i \in \mathcal{D}$ do                    ▷ Assignments for Coverage
4:      $min\_distance = \infty$
5:      $best\_cloudlet = \emptyset$
6:      for $m_j \in \mathcal{M}$ do
7:          $c_k$ = CloudletCell(MCP, $m_j$)
8:          $distance = \Delta(c_i^t, c_k)$
9:          if $\mu_j^t \geq \delta_i^t$ and $distance \leq r_j$ then
10:             if $distance < min\_distance$ then
11:                 $best\_cloudlet = m_j$
12:                 $min\_distance = distance$
13:     if $best\_cloudlet \neq \emptyset$ then
14:         $\mu_j^t = \mu_j^t - \delta_i^t$
15:         DCA $\leftarrow \{m_j, d_i\}$
16:     else
17:         $uncovered\_devices \leftarrow d_i$
18: for $d_i \in uncovered\_devices$ do     ▷ Assignments for Mobility
19:     for $m_j \in \mathcal{M}$ do
20:         $distance = \Delta(c_i^t, c_k)$
21:         if $distance \leq 2 * r_j$ then
22:             DCA $\leftarrow \{m_j, d_i\}$
23: Output: DCA
```

We define $\overrightarrow{\eta} = \Delta(\lambda_i^t, w_j(t-1))$ as the displacement between the position of mobile cloudlet m_j at $t-1$ to the current device position λ_i^t. Hence, force $\overrightarrow{F_j}(t)$ denotes the expected weighted positional change of mobile cloudlet m_j. Note that force $\overrightarrow{F_j}(t)$ is a vector quantity with component forces in the direction of each 3D axis based on the displacement vector $\overrightarrow{\eta}$.

Since the mobility of the cloudlets is constrained by the maximum travel parameter Λ_j, we update the force value. The actual value of the force used in deducing the new position is given by Equation (17).

$$\overrightarrow{F_j}(t) = \begin{cases} \Lambda_j & \text{if } \overrightarrow{F_j}(t) > \Lambda_j \\ \overrightarrow{F_j}(t) & \text{otherwise.} \end{cases} \tag{17}$$

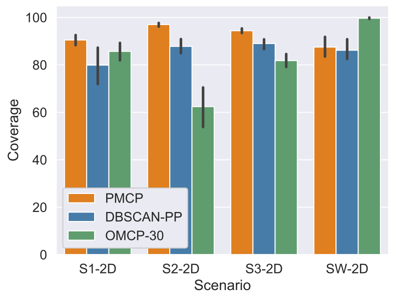
Then, the estimated position $\widehat{w_j}(t)$ of mobile cloudlet m_j is given by Equation (18).

$$\widehat{w_j}(t) = \overrightarrow{F_j}(t) + w_j(t-1), \tag{18}$$

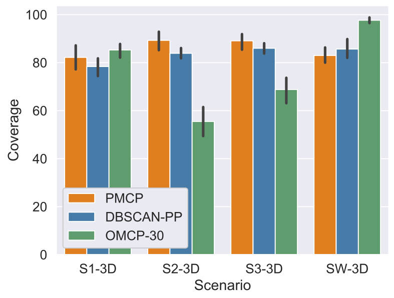
where the vector components of the force are added to the corresponding 3D coordinates of the position of the cloudlet in the previous time slot $w_j(t-1)$. Note that MCP from the initial placement (Algorithm 1) is utilized as $w_j(1)$.

After these calculations (lines 6-8), the final position $w_j(t)$ of the cloudlet is determined by finding the closest cell c_k to the estimated position $\widehat{w_j}(t)$ (line 9). Here, only the cells that are not already assigned to a previously evaluated mobile cloudlet are considered for the final position. The history of cloudlet locations is maintained in MCP(t), and the device assignments DCA are updated for use in the next round (lines 10-11). Every cloudlet moves to its new final position $w_j(t)$, i.e., the center of the cell c_k using MCP(t).

To summarize, the mobile cloudlets move towards the devices with the highest attractive force and finally relocate



(a) Coverage for 2D Scenarios



(b) Coverage for 3D Scenarios

Fig. 3: Comparison of coverage values at each time slot.

to the centers of the cells, closest to their estimated positions in the grid. Algorithm 2 outlines the entire process of the adaptive placement of PMCP. After each adaptive placement iteration, the devices are individually mapped to the relocated mobile cloudlets using Algorithm 3, and the mappings are used to make the placement decision in the next round.

V. Experimental Results

A. *Experimental Setup*

For an extensive evaluation, the experiments on the PMCP approach need to be performed under a realistic user mobility model and appropriate distribution of changing user demands. We utilize the KTH Walkers dataset [12] to establish mobility scenarios for our experiments. The dataset represents traces of pedestrian mobility from a part of downtown Stockholm. Moreover, the dataset was specifically designed to study mobility models for opportunistic communications, making it a highly suitable dataset for evaluating our approach. There are four sub-traces (3 Olstermann traces and 1 Subway trace) in the KTH Walkers dataset, each of which we use to generate one 2D and one 3D mobility dataset for our experiments. We likewise assume a uniform distribution for mobile cloudlet capacities and individual user demands. We experiment with a different number of devices to test the scalability of our approach and deduce the number of mobile cloudlets for each scenario based on the number of devices.

We compare the performance of the PMCP with the optimal solutions by OMCP and the results of DBSCAN with path planning (DBSCAN-PP). DBSCAN-PP is chosen as a benchmark since it is a well-known density-based clustering method and behaves similar to existing mobile cloudlet placement approaches in the literature (e.g., [26], [28]) when we add the path planning component. It is hence a strong benchmark given the unavailability of implementation of more recent studies.

TABLE I: Experiment Scenarios

Scenario	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert T\rvert$	μ_j^t	δ_i^t
Scenario3 (S3-2D/3D)	16	300	10	$U[140, 170]$	$U[2, 10]$
Scenario2 (S2-2D/3D)	12	180	10	$U[120, 160]$	$U[2, 10]$
Scenario1 (S1-2D/3D)	8	100	10	$U[80, 100]$	$U[2, 10]$
Subway (SW-2D/3D)	8	100	6	$U[80, 120]$	$U[2, 10]$

We compare these approaches in terms of user coverage attained at each time slot, the rate of cloudlet switches made by devices (to test the stability of each approach), cumulative distance traveled by the cloudlets, and the running time across different time slots. We also visualize the cloudlet placements to demonstrate the movement of the mobile cloudlets in response to dynamic changes in device specifications.

Table I presents the different experiment scenarios in terms of the number of mobile cloudlets available, the number of devices, the number of time slots in the experiment, distribution of cloudlet capacities, and distribution of device demands. For each scenario, both 2D and 3D mobility datasets have been created. 2D datasets are direct representations of available traces, while 3D datasets were generated by assuming a 3D landscape over which mobility happens. This is done by adding non-negative z-coordinates (to denote elevations of the assumed landscape) to the existing trace data. In addition, the maximum travel parameter Λ_j for all of these experiment scenarios are based on a uniform distribution $U[2, 3]$.

We obtain the optimal results from OMCP by solving the integer program using IBM ILOG CPLEX Concert Technology API for Java [35]. All approaches are implemented in the same version of Java, and all experiments are run on the same JVM on the Nautilus HyperCluster [36] with 16 CPU cores and 64 GB RAM. This justifies a direct comparison between the approaches. It is also noteworthy that OMCP is an NP-hard problem. As such, CPLEX may never converge to a provably optimal result for some experiment scenarios. Hence, the best results obtained within a specified time limit (30 minutes) are presented for consistency and are shown by OMCP-30. This means, if OMCP could not solve the problem optimally in 30 min, the best obtained result in this duration has been used for comparison. Note that optimizing more than 30 minutes showed no significant improvement in our results of all experiment scenarios.

B. *Analysis of Results*

1) Coverage: We first compare the coverage values of PMCP, DBSCAN-PP, and OMCP-30. Figure 3a shows the mean and standard deviations of the 2D scenarios. The mean value of coverage obtained by PMCP across different 2D

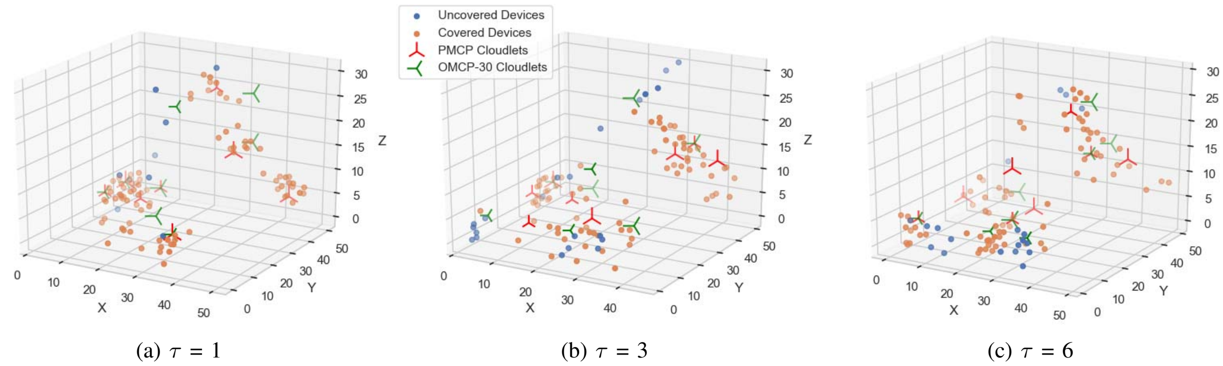


Fig. 4: Visualization of comparative cloudlet placements and PMCP coverage at different time slots.

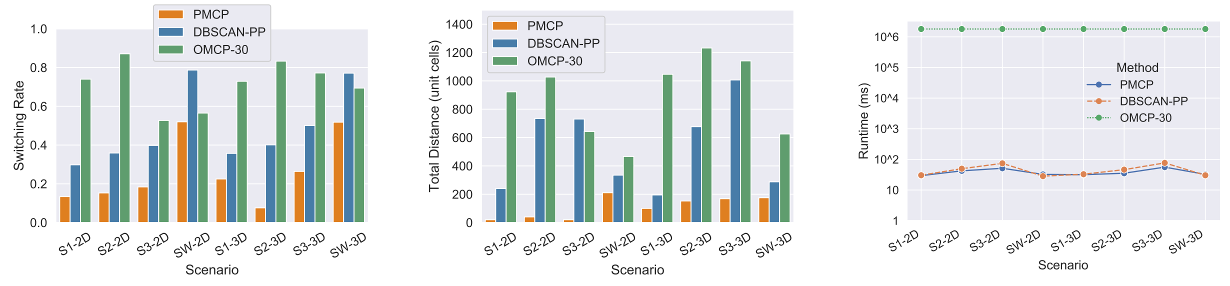
scenarios and time slots is 92.89% with standard deviation of 4.63. In all scenarios, PMCP outperforms DBSACN-PP in terms of coverage. DBSCAN-PP obtains lower mean coverage of 85.67% with standard deviation of 8.3. As an NP-hard problem, OMCP does not guarantee termination with an optimal solution in a finite time. As the problem size increases in terms of the number of devices and time slots, it is more difficult to obtain the optimal solutions or even high-coverage time-limited solutions using OMCP-30. This is most noticeable in Scenario 2 (S2-2D and S2-3D).

Figure 3b shows the coverage results of the 3D scenarios. The mean coverage obtained by PMCP is lower than the 2D scenarios but still high at 86.22% with standard deviation of 7.14. DBSCAN-PP is more consistent with a lower standard deviation of 5.59, however, it again achieves lower mean coverage of 83.25%. The coverage values obtained by PMCP are consistently more than DBSCAN-PP and OMCP-30 with lower standard deviations except for a few cases as seen in Figure 3. The Subway scenarios (SW-2D and SW-3D) and Scenario1-3D (S1-3D) are exceptions to this observation where OMCP-30 has higher coverage with a lower standard deviation. This is because the size of these scenarios is much smaller. They either have a smaller number of devices or fewer time slots or both. Hence, it is easier for an exhaustive approach like OMCP-30 to find values closer to the optimal solution. DBSCAN-PP performs better in terms of coverage in Subway-3D (SW-3D) scenario since this scenario has distinct dense clusters, but PMCP is still comparable given a large overlap in the error bars of both approaches. Moreover, cloudlets can move more freely in DBSCAN-PP since new cluster centers can be far (larger than Λ_j) from current cloudlet cells, which may lead to infeasible placements.

2) Adaptive Nature: We visualize the cloudlet placements to illustrate the smooth adaptive movement of cloudlets in PMCP in response to the devices' movements. In Figure 4, we present the comparative cloudlet placements according to PMCP (red) and OMCP-30 (green) along with the devices (blue and orange based on PMCP coverage) across different time slots of Subway 3D (SW-3D) scenario. The device distributions across time slots first show that Subway 3D scenario realistically depicts the visible 3D features in subway stations like stairs and escalators at the entrance/exit and the area around and outside the station. Figure 4a shows the initial placement at $\tau = 1$, which looks better for PMCP since it places cloudlets (red) close to the demand centers (with coverage of 91%). Hence, the cloudlet placements by PMCP appear closer on average to the devices than the placements by OMCP-30 (green). There are some exact overlaps between PMCP cloudlets and OMCP-30 cloudlets too. In comparison, OMCP-30 cloudlets are placed more strategically to cover a larger number devices (about 95%) but are not close to the devices in some of their locations.

As the devices move, both approaches readjust the placements of the cloudlets. At $\tau = 3$ (see 4b), the placements by OMCP-30 look better based on the distribution of devices. The coverage of OMCP-30 here is 96% compared to only 78% for PMCP. However, PMCP has significantly readjusted cloudlet placements based on the new user distribution. As the devices have scattered more, so have the PMCP cloudlets. Moving into final time slot $\tau = 6$, Figure 4c shows that PMCP has caught up significantly in terms of following the devices and has placed cloudlets close to device clusters at the bottom-left and the top-right (notice the wider coverage in the figure), which did not have nearby PMCP cloudlets in $\tau = 3$. The visualizations here validate that PMCP places the mobile cloudlets by truly responding to dynamic and uncertain changes in device distribution at every time slot and achieves consistent coverage compared to the optimal coverage. They also demonstrate that PMCP does not favor the devices in dense regions only and adapts to even sparse regions over time. PMCP fairly serves all devices in the long-term since the devices far away from the dense regions are eventually covered too.

3) Stability: Providing stable services is also critical as it reduces excessive migration that will lead to high latency. Both high coverage and stable service are the motivations behind the design of our approach. As a result, our approach compensates for the coverage gap by improving the stability of the services. We measure the stability in terms of the *switching rate* for devices. The switching rate is the total number of cloudlets switched by the devices between consecutive time slots divided by the total number of connections established by the devices across all time slots. Thus, a lower switching rate means the



(a) Cloudlet switching rate (b) Cloudlet distance traveled (cell units) (c) Total running time (ms)

Fig. 5: Comparing cloudlet switching rate of devices, cloudlet distance traveled, and running time.

devices are likely to be served by the same cloudlet across consecutive time slots leading to fewer migrations between cloudlets and ultimately smoother, more stable edge services. Figure 5a shows the switching rate of the three approaches across all experiment scenarios.

The average switching rates of PMCP, DBSCAN-PP, and OMCP-30 are 0.259, 0.484, and 0.716, respectively. This means only 25.9% of the connected devices change cloudlets between the two time slots in PMCP while a more significant number of devices 48.4% and 71.6% do that in DBSCAN-PP and OMCP-30, respectively. The high rate obtained by OMCP-30 is because it is agnostic to switching and objectively focuses on maximizing coverage, while DBSCAN-PP has path planning to minimize the movement of cloudlets between clusters which offers some advantage. Hence, even in the higher coverage scenarios, users using DBSCAN-PP and OMCP-30 experience more delays in their services. Though we can make OMCP-30 more adept at handling switching, introducing hard constraints to limit switching in OMCP-30 may even render some scenarios theoretically infeasible to solve. This further supports the performance of PMCP in terms of coverage since OMCP-30's coverage values are extremely high benchmarks to compare against. Among all scenarios, the only comparable switching rate between the PMCP and OMCP-30 can be noticed for SW-2D scenario. This indicates that OMCP-30 probably found a solution with less switching by chance. Otherwise, PMCP always has a significantly lower switching rate, consequently, ultra-low latency services, compared to both DBSCAN-PP and OMCP-30.

4) Energy Efficiency: Given that mobile cloudlets are deployed for the same distribution of devices and assuming all other factors are constant, the energy consumption of these mobile cloudlets largely depends on their total distance traveled. Hence, we assess the energy efficiency using the cumulative distance traveled by the cloudlets in terms of unit cells, i.e., the distance between centroids of the unit cells switched by the cloudlets across all time slots. As shown in Figure 5b, PMCP is extremely efficient in terms of cloudlet movements for all scenarios. The mean cumulative distance traveled by the cloudlets are 110.84, 525.69, and 887.95 for PMCP, DBSCAN-PP, and OMCP-30, respectively across all scenarios. For some scenarios, the cloudlets have negligible movements between cells in PMCP, notably S1-2D and S3-2D. This is because in PMCP, cloudlets do not move to a different cell unless there is a significant change in the user distribution in and around their covered cells. DBSCAN-PP is significantly worse since it re-clusters the devices in each time slot and moves cloudlets with no maximum travel constraint to newly established cluster centers based on matching. Unrestricted movement with poorer energy efficiency makes DBSCAN-PP impractical. These are obvious limitations of the existing approaches in the literature that are based on similar principles. OMCP-30 is again agnostic to minimizing the distance traveled and purely focuses on maximizing coverage.

5) Running Time: We compare the running time of PMCP against DBSCAN-PP and OMCP-30 (i.e., 30-minute time limit shown by the flat line). Illustrated by Figure 5c, PMCP has exceptionally low running time compared to OMCP-30. DBSCAN-PP has a similar running time, however, DBSCAN needs to be tuned using two hyperparameters (ϵ and $minPoints$), and the running time shown here is for a tuned version. If there are n combinations of hyperparameters considered for tuning DBSCAN, then obtaining the presented results would take n times the presented running time. On the contrary, PMCP runs in a few milliseconds with total running time for any scenario never exceeding 56 milliseconds. Most of this time is spent on the initial placement of the mobile cloudlets, where mobility decisions are not yet being made. Also, this figure shows the sum of the running times across all time slots (sum of 6-10 time slots). Calculations in the adaptive part of the approach only take close to 1-3 milliseconds for each time slot, even for the largest experimental scenario.

In summary, PMCP offers consistent high device coverage, stable services through low switching rate, and markedly superior energy efficiency through reduced cumulative distance traveled by the cloudlets. PMCP is well designed to run in real-time and scales well across scenarios.

VI. Conclusion

Deployment of mobile cloudlets is an elegant solution to provide better services in next-generation edge networks with high user mobility and dynamic, uncertain demands. To address the goals of enhancing user coverage and providing stable edge services, we designed a novel dynamic mobile cloudlet placement approach, PMCP, inspired by the concepts in physics. The results show that PMCP not only achieves high coverage and more stable services but is also energy-efficient and runs in real-time. This means our approach is practical for dynamic cloudlet placements and can also be extended to solve dynamic assignment and resource placement problems. Since the approach includes iterative calculations, it can also be parallelized to run faster. We plan to study the distributed decision making for dynamic cloudlet placements next.

Acknowledgment

This research was supported in part by Cisco grant CG#1935382 and NSF grant CNS-2145268. We acknowledge the support from the National Research Platform (NRP) for providing computing resources for our experiments through their Nautilus HyperCluster.

References

[1] S. Tang, B. Chen, H. Iwen, J. Hirsch, S. Fu, Q. Yang, P. Palacharla, N. Wang, X. Wang, and W. Shi, "VECFrame: A vehicular edge computing framework for connected autonomous vehicles," in *Proc. of the IEEE International Conference on Edge Computing*, pp. 68–77, 2021.

[2] E. F. Maleki, W. Ma, L. Mashayekhy, and H. La Roche, "QoS-aware 5G component selection for content delivery in multi-access edge computing," in *Proc. of IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 1–10, 2021.

[3] P. Merz and M. Rylander, "How 5G-Advanced will help quench the world's growing thirst for capacity." [Online] https://www.nokia.com/blog/how-5g-advanced-will-help-quench-the-worlds-growing-thirst-for-capacity/, June 2021.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[5] L. Miller and J. Cavazos, *5G & Beyond For Dummies*. Wiley, 2022.

[6] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, 2020.

[7] M. Moradi, K. Sundaresan, E. Chai, S. Rangarajan, and Z. M. Mao, "SkyCore: Moving core to the edge for untethered and reliable UAV-based LTE networks," in *Proc. of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 35–49, 2018.

[8] D. Patel, "FAA teams up with Verizon to test drones connected to mobile phones." [Online] https://industry-update.com/faa-teams-up-with-verizon-to-test-drones-connected-to-mobile-phones-general-aviation-news/74001/, 2021.

[9] M. K. Shehzad, S. A. Hassan, M. Luque-Nieto, J. Poncela, and H. Jung, "Energy efficient placement of UAVs in wireless backhaul networks," in *Proc. of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*, pp. 1–6, 2020.

[10] M. Mozaffari, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, "Beyond 5G with UAVs: Foundations of a 3D wireless cellular network," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 357–372, 2018.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, pp. 226–231, 1996.

[12] S. T. Kouyoumdjieva, O. R. Helgason, and G. Karlsson, "Microsimulation of pedestrian mobility, CRAWDAD dataset." https://crawdad.org/kth/walkers/20140505, 2014.

[13] S. Kang, L. Ruan, S. Guo, W. Li, and X. Qiu, "Geographic clustering based mobile edge computing resource allocation optimization mechanism," in *Proc. of the 15th International Conference on Network and Service Management*, pp. 1–5, 2019.

[14] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2017.

[15] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, pp. 1–21, 2019.

[16] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 16, pp. 1–9, 2017.

[17] Z. Wang, F. Gao, and X. Jin, "Optimal deployment of cloudlets based on cost and latency in internet of things networks," *Wireless Networks*, vol. 26, no. 8, pp. 6077–6093, 2020.

[18] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. of the 2018 IEEE International Conference on Edge Computing*, pp. 66–73, 2018.

[19] F. Wang, X. Huang, H. Nian, Q. He, Y. Yang, and C. Zhang, "Cost-effective edge server placement in edge computing," in *Proc. of the 5th International Conference on Systems, Control and Communications*, pp. 6–10, 2019.

[20] D. Lu, Y. Qu, F. Wu, H. Dai, C. Dong, and G. Chen, "Robust server placement for edge computing," in *Proc. of the IEEE International Parallel and Distributed Processing Symposium*, pp. 285–294, 2020.

[21] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 926–937, 2019.

[22] D. Bhatta and L. Mashayekhy, "Generalized cost-aware cloudlet placement for vehicular edge computing systems," in *Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 159–166, 2019.

[23] D. Bhatta and L. Mashayekhy, "A bifactor approximation algorithm for cloudlet placement in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1787–1798, 2022.

[24] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 6, pp. 621–655, June 2008.

[25] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, M. H. Tareque, and M. S. Hossain, "Routing in flying ad hoc networks: survey, constraints, and future challenge perspectives," *IEEE Access*, vol. 7, pp. 81057–81105, 2019.

[26] H. Xiang, X. Xu, H. Zheng, S. Li, T. Wu, W. Dou, and S. Yu, "An adaptive cloudlet placement method for mobile applications over GPS big data," in *Proc. of the IEEE Global Communications Conference*, pp. 1–6, 2016.

[27] Y. Zhang, K. Wang, Y. Zhou, and Q. He, "Enhanced adaptive cloudlet placement approach for mobile application on spark," *Security and Communication Networks*, vol. 2018, pp. 1–12, 2018.

[28] X. Jin, F. Gao, Z. Wang, and Y. Chen, "Optimal deployment of mobile cloudlets for mobile applications in edge computing," *The Journal of Supercomputing*, vol. 78, no. 6, pp. 7888–7907, 2022.

[29] J. Wang, K. Liu, and J. Pan, "Online UAV-mounted edge server dispatching for mobile-to-mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1375–1386, 2020.

[30] X. Yuan, M. Sun, and W. Lou, "A dynamic deep-learning-based virtual edge node placement scheme for edge cloud systems in mobile environment," *IEEE Transactions on Cloud Computing (Early Access)*, pp. 1–12, 2020.

[31] R. H. Ballou, "Potential error in the center of gravity approach to facility location," *Transportation Journal*, pp. 44–50, 1973.

[32] W. E. Wright, "Gravitational clustering," *Pattern recognition*, vol. 9, no. 3, pp. 151–166, 1977.

[33] P. Binder, M. Muma, and A. M. Zoubir, "Gravitational clustering: a simple, robust and adaptive approach for distributed networks," *Signal Processing*, vol. 149, pp. 36–48, 2018.

[34] T. Wang, L. Qiu, A. K. Sangaiah, A. Liu, M. Z. A. Bhuiyan, and Y. Ma, "Edge-computing-based trustworthy data collection model in the internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4218–4227, 2020.

[35] IBM, "Overview (CPLEX Java API Reference Manual)." [Online] https://www.ibm.com/docs/en/icos/12.10.0?topic=v12100-introduction.

[36] Pacific Research Platform, "Nautilus documentation." [Online] https://ucsd-prp.gitlab.io/nautilus/.