IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 13, 2020, accepted September 24, 2020, date of publication September 29, 2020,
date of current version October 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027707

Cognitive Caching at the Edges for Mobile Social Community Networks: A Multi-Agent Deep Reinforcement Learning Approach

MILENA RADENKOVIC AND VU SAN HA HUYNH
School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, U.K.
Corresponding author: Milena Radenkovic (milena.radenkovic@nottingham.ac.uk)

ABSTRACT Content caching in the current commercial content delivery networks (CDNs) allows reduction of duplicate traffic and improvement of QoS and QoE but it still suffers from surges of content traffic, network congestion, high mobility of users and dynamic users' content request patterns which may result in high content access latency. With the increasing interest of large companies in providing next-generation mobile edge applications and services that the users can use despite potentially sparse, non-uniform connectivity, it is becoming increasingly important to provide efficient, smart content caching services at the edges to help with scalable storage and processing of local data as well as sharing data both at the edges and in the cloud. We propose a novel multi-agent deep reinforcement learning approach, CognitiveCache, in which edges adaptively learn their best caching policies while collaborating with other neighbouring edges to better understand if they can be usable for cache content placement optimisation problem in dynamic environments. We show that CognitiveCache can respond and adapt to the spatial-temporal locality of dynamically changing content workloads and resources, improve the reliability and scalability of content sharing, enhance QoE for users and decrease operational costs in mobile social community networks. We perform extensive multi-criteria evaluation of our proposal against four benchmark and competitive protocols over two different real-world scenarios in New York and London in the face of different mobility and users' interest patterns to show that CognitiveCache achieves higher cache hit ratios, lower delays while reducing resource consumption.

INDEX TERMS Mobile social community networks, edge cloud and fog networks, content caching, deep reinforcement learning, multilayer spatial-temporal locality.

I. INTRODUCTION

Current large-scale networking systems have been evolving to adapt to the increasing complexity and dynamics of both underlying networking infrastructures and applications. Content caching in content delivery networks (CDNs) [63], [64], such as AWS Cloudfront [25] and Azure CDN [61], allows improvement of users QoS and QoE but it still suffers from sparse network coverage, disconnections, network congestion and highly dynamic users' mobility and query patterns [1], [2], [6]. Many applications, such as remote health care and mobile social networks, need to be supported by next-generation mobile edge predictive content services which allow localized content storing and processing close to the users interested in it [1], [6], [7], [32]–[34].

The associate editor coordinating the review of this manuscript and approving it for publication was Miguel López-Benítez.

State-of-the-art edge services hosted in the mobile edge devices bring local data management, computation and inference capabilities to the edges to reduce the delay and improve the performance of data transport for end-users [67]. However, they still have limited support for surges of traffic, especially video streaming and dynamically changing networks due to users' mobility and content request patterns [1], [2]. To enable fully local network, interest and privacy awareness self-organised multi-layer cognitive edge clouds have been proposed in [32], [34] to host various services. Edge and fog computing [3], [41], [50] integration with various technologies including device-to-device (D2D) communication [30], [43], content-centric architecture [1], [11], small cells [48], caching [1]–[3], [8] are proposed to support complex networking data services. Intelligent caching services at the edges are envisaged to be important solution providing more localised and more responsive content

services to mobile users compared to the traditional CDN approaches.

We formulise the objective function of cache content placement optimization problem in dynamic environment and investigate if machine learning (ML) – reinforcement learning (RL) approaches can be helpful for caching services that are able to compute cache content placement among the edges such that the aggregate benefit is maximized. RL algorithms are traditionally used in statistics, neuroscience and have been successfully applied in a variety of applications ranging from robotics [49], energy and resource management [22], [50], recommendation systems [19], [51], transportation [52] and software-defined networks [21]. Reinforcement learning [15], [16] is suitable for our unreliable network environments with no global knowledge and a certain degree of dynamics where we can quantify different variables and states of the network environments. We argue that application of RL's core concept where a mobile edge learns behaviour through ''trial-and-error'' interactions in a dynamic environment is feasible in the considered scenario (even though it is not for other scenarios such as e.g. high speed police chase which we do not consider in this paper).

In this paper, we propose a multi-agent deep reinforcement learning (DRL) model for caching framework CognitiveCache at the edges that adaptively and collaboratively capture and predict the dynamic changing networks and complex users' content request patterns to improve the accuracy of caching decision to place the most suitable contents in the most suitable edges. We model mobile edge-cloud network environment with a novel state space and set of actions that edges could take to collaborate with other nodes, interact with the environment in real time to maximise a cumulative reward. Traditional single-agent DRL-based caching approaches [54], [55] have been proposed where a single edge (e.g. single base-station or an access-point) learns to make most suitable caching decisions based on the states of the environment and the rewards. In our context with multiple edges (e.g. femtocell, access points, mobile users, vehicles, etc.), single-agent DRL-based caching approaches are not applicable due to i) every individual edge should learn its own caching policy relied on its observed user content request patterns and the current state of its caching storage ii) single central agent could have limited scalability as the increasing number of distributed edges will result at massive action space [56]. Therefore, we investigate a DRL caching approach for mobile edge-cloud scenarios where multiple edges can collaborate with each other. We utilise and extend the actor-critic based approach [5], [57] for multi-agent reinforcement learning [18], [24] in which the actor network controls the caching decisions and the critic network evaluates and gives feedback on the chosen ones. In CognitiveCache, each edge considers its caching strategy together with the caching policy of its neighbouring edges as part of the environment [38]. This implies that all edges have some effect on the environment and similar action of a single edge can have different outcomes depending on what the other edges are doing. We consider mobile social community networks which are characterized by multi-network layer [1], [3], [29], [45] including physical geo-temporal network topologies, resources, social geo-temporal communities and content geo-temporal interests. Note that in this paper, our edges are assumed to be privacy-aware by design as proposed in [32] and [34]. Malicious behaviour between the edges is out of the scope of our paper.

We show that CognitiveCache outperforms benchmark and state of the art caching protocols across a range of metrics to achieve higher cache hit ratio, lower latency and lower transmission cost compared in the face of realistic dynamic users' mobility and data demand patterns. We drive our experiments by real-world mobility traces and users' interest traces of more than 10^3 mobile edges and 10^5 content interests from New York Foursquare & London Twitter datasets. The remaining of the paper is organized as follows. Section 2 provides a systematic overview of the related work. Section 3 describes key features of our multi-agent deep reinforcement learning caching framework and provides pseudo-code. Section 4 discusses multi-criteria evaluation of CognitiveCache against other benchmark and competitive protocols across a range of metrics for realistic dynamic users' mobility and data demand patterns. Section 5 gives conclusions and discusses future work.

II. RELATED WORK

Machine learning [15], [16], a subset of artificial intelligence, has attracted increasing attention of scientific communities in recent years as it allows to perform a specific task without explicit instructions. Machine learning approaches can be grouped into three fundamental paradigms [15], [16]: (1) supervised learning that infers a learning function from labelled data; (2) unsupervised learning that involves mining information and learning from unlabelled or raw data; (3) reinforcement learning (RL) that explores how agents take suitable actions to maximise rewards in an environment. The environment is typically stated in the form of a Markov decision process [15], [46]. In reinforcement learning [15], [16], given a goal, an agent learns how to achieve the goal by trial-and-error interactive process with its environments. Both supervised and unsupervised learning are more suitable for off-line learning scenarios since all the inputs are received at once while reinforcement learning can be classified as online learning algorithms as it relies on being able to continuously monitor the response of the actions taken, and measure against a definition of a reward [16]. One of the well-known reinforcement learning algorithms is Q-learning [17], in which an agent in a given state estimates the expected reward if it chooses a specific action and enters another state. The core of the algorithm is a Bellman value iteration update [17] which is used to find optional action-selection strategies. Authors in [67] argue that to reduce the delay in data processing and minimize the privacy risks of revealing raw data to service providers, future AI and machine learning-based services could be deployed

on users' devices at the edges rather than placing everything on the cloud although moving ML-based data analytics from cloud to the edge devices brings a series of challenges.

In recent years, different deep reinforcement learning based caching approaches at the network edges have been proposed leveraging its powerful learning capacity from the historical events. Authors in [54] propose a DRL based framework for content caching at the base station as a single agent. The authors utilize actor-critic method and train the policy using Deep Deterministic Policy Gradient (DDPG) [58] to improve the caching decision, maximise the long-term cache hit rate without knowledge of content popularity distribution. However, [54] is mostly confined to centralized learning within one base station as a single agent which is not scalable in mobile edge-cloud scenarios [56]. Authors in [55] propose a deep reinforcement learning-based caching in hierarchical content delivery networks. The proposed framework DQNCache relies on Deep Q Networks to learn optimal caching policy in an online manner. DQNCache tries to find optimal value function which is a mapping between an action and a value to find better action with higher value. DQNCache belongs to value-based algorithm which is different from policy-based algorithm such as REINFORCE [59]. While value-based approaches are more sample, efficient and steady, policy-based approaches are better for continuous, stochastic environments and have a faster convergence. In this paper, we utilize actor-critic method [5], [57] which takes advantage of both value-based and policy-based algorithms while eliminating their drawbacks. Similar to [54], [55] is built on single-agent DRL approach where a parent node makes caching decisions by observing aggregated requests from all leaf nodes. Different from existing single-agent DRL approaches [54], [55], our CognitiveCache leverages multiagent DRL to predict and adapt in real time the dynamically changing spatial-temporal locality of networks and users requests [3], [4] and achieve collaborative intelligence between edge caching in mobile edge-cloud networks. Each CognitiveCache edge has its own learning model and exchanges knowledge with its neighbours to provide better scalability compared to centralised DRL caching approaches. Authors in [56] propose a multiagent DRL based caching for video contents which leverages independent Q-learning, advanced actor-critic method integrated with long short-term memory network to adaptively learn most optimal local caching policy in conjunction with other edges. In this paper, we propose to use soft actor-critic method [5], [57] which promises to be more sample efficient and more robust to brittleness in convergence compared to [56]. CognitiveCache has a fine-grained state design that allows CognitiveCache to adaptively capture and forecast the popularity of content and surge of content demand.

Reinforcement learning has also been applied in different areas of mobile edge cloud networks. Authors in [37] propose a delay-tolerant congestion-control framework that automatically modifies its operations based on the dynamic changings of the underlying network using reinforcement Q-learning. [37] shows that it continuously adapts to the dynamics of the target environment in a variety of DTN applications and scenarios with adequate performance. However, [37] only takes into account resource availability as states of nodes without considering other dimensionalities such as connectivity and content traffic, thus it does not sufficiently represent the multilayer multidimensional characteristics of mobile network environments. Authors in [68] propose a distributed and energy-efficient control framework based on convolutional neural network (CNN) that utilizes distributed multi-agent DRL approach to solve a challenge of mobile crowdsensing. Reference [68] takes energy availability, spatial coordinates and remaining data as states for each node in the network. The solution explores the spatio-temporal nature of the considered scenario for better cooperation and competition between nodes to maximize the data collection ratio, geographic fairness and energy efficiency. Authors in [19] leverage the deep learning techniques for sequential modelling and correlation identification of user interests influenced by their social circles and centrality. Reference [19] shows that they significantly improve prediction accuracy to predict user interests based on their sociality compared to widely used baseline methods. However, [19] assumes centralized knowledge and does not support fully distributed decision making as we do in our paper. Authors in [20], [21] utilize deep reinforcement Q-learning to propose an integrated framework that can enable dynamic orchestration of not only networking but also caching and computing resources in order to improve the performance of next-generation vehicular networks. However, the framework's complexity is very high when the network states, caching states and computational resource states are jointly considered. Authors in [22] propose an intelligent deep reinforcement learning-based offloading mechanism for vehicular edge where the states of communication, mobility and computation are modelled by finite Markov chains. Similar to [20], [21], task scheduling and resource allocation strategy is formulated as a joint optimization problem to maximize users' QoE.

Many existing predictive analytics, collaborative heuristics and utility-based mobile edge caching models have been proposed in recent years. Research in [1] models mobile edge-cloud networks as a bargaining game theory to propose a collaborative adaptive caching framework CafRepCache. The authors formulate content discovery, caching and retrieval as the optimization problem and prove it is a typical Integer Programming program which is NP-Complete [1], [6]. CafRepCache serves subscribers with its local cache or by redirecting a request to a nearby collaborative cache, rather than forwarding to the original publisher. CafRepCache is built on multilayer predictive analytics and heuristics to capture and predict content interests coming from dynamic changing clusters [1]–[3], [14], [31], [39] of subscribers in both random and scale-free network, and thus reduce content retrieval delay, improve cache efficiency and reduce resource consumption while enabling responsiveness to heterogeneous dynamically changing network

topology, congestion avoidance and varying patterns of content publishers/subscribers. Authors in [30] present InfluentialCache, a proactive caching approach in small cellular device-to-device communication networks. InfluentialCache models D2D cellular network as a social graph in order to utilize its spatial structure. The influential users and their clusters are identified using eigenvector centrality and CC-GA clustering algorithms [42]. The authors assume that the content requested, generated, or accessed by the influential users in a community will become popular within this community, thus proactively cache this content to serve the others. Authors in [8] propose SocialCache, a caching algorithm based on social relationship of nodes in the network to choose caching carriers. Content popularity is calculated based on the frequency and freshness of content requests. Authors in [12] propose LocationCache that uses function of distances between subscribers and caching points and other features such as time stamp or number of content requests to classify contents and replace them when cache memory is full. Least frequently used (LFU) [10] measures the frequency of content requests at a caching node to make caching decisions. The content receiving more frequent requests will need to be cached because it will have higher chance of being requested again in the future. Least recently used (LRU) [9] records the time stamp of a content request locally in order to make caching decisions based on how recent the content requests are. ProbCache [60] keeps a copy of content in a cache along a path with a probability which is calculated based on path lengths and multiplexes content flows. In this paper, we compare CognitiveCache against state-of-the-art and benchmarking caching algorithm including the least recently used (LRU) [9], least frequently used (LFU) [10], ProbCache [60] and DQNCache [55].

In previous works, we have proposed and deployed fully-distributed real-time multi-layer mobile edge cloud architectures for enabling multiple services for smart vehicles, drones, cities and agriculture applications spanning MODiToNeS [32], mobile personal edge-clouds [66] and Raspberry PI based personal clouds RasPiPCloud [32]–[34] which support multiple on-demand virtual containers (e.g. LXC, Docker) to host different services and applications that collect, store, analyse, predict and share data with other edges while retaining completed control and ownership of their data.

III. COGNITIVECACHE – A MULTI-AGENT DEEP REINFORCEMENT LEARNING BASED CACHING FRAMEWORK

A. COGNITIVECACHE FRAMEWORK AND SYSTEM MODEL

We envisage a distributed edge caching scenario for heterogeneous content services such as video streaming, file downloading, used by dynamic groups of mobile users who request contents in real time. In this context, multiple CognitiveCache edges (e.g. femtocell, access points, mobile users, vehicles, etc.) are located in different areas providing processing and caching capacity. For example, a group of students in a city area acting as subscribers send their interests of information about nearby parking space, local coffee shops or interests of bus schedule to CognitiveCache edges. CognitiveCache edges can communicate with other neighbouring edges and can collaboratively retrieve the requested contents from each other (Figure 1).

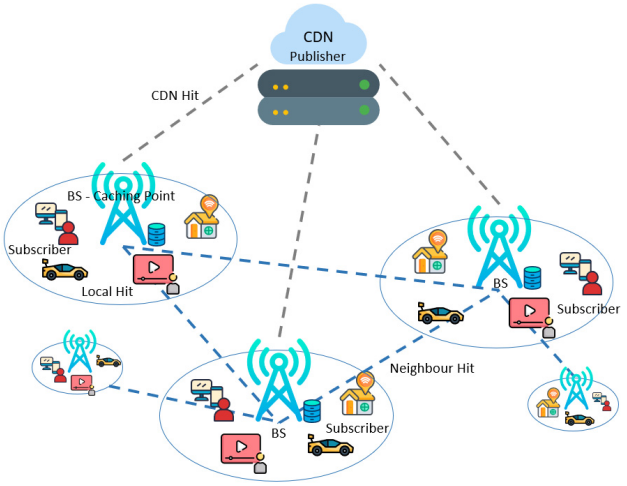


FIGURE 1. Mobile edge caching in social edge-cloud community networks.

While the edges have limited caching storage and thus can only cache a certain amount of contents, CDN server is assumed to have sufficient caching capacity with all the requested contents already being cached and is able to provide requested contents to the users. However, we assume that retrieving contents from CDN server has a significantly higher delay compared to retrieving contents from the edges (e.g. BSs). More specifically, we propose that serving a request has three phases: local edge cache hit, neighbour edge cache hit and CDN cache hit. Local edge cache hit: when a request arrives at the local edge, it sends the cached content to users if the requested content is found in its local caching storage. Neighbour edge cache hit: when a local edge does not cache the requested content, it then attempts to retrieve the content from its neighbour edges which cause extra latency but is still quicker and cost-beneficial compared to retrieving content from the CDN server. CDN cache hit: when the requested contents cannot be found from either local or neighbour edges' cache storage, local edge fetches the requested contents from CDN server as a lowest-priority back-up solution. We assume the CDN's latency is the same for all users.

We propose a multi-agent deep reinforcement learning (DRL) model for caching framework CognitiveCache at the edges with novel design of states, actions and rewards. We utilise independent Q-learning [38] approach to solve the multi-agent reinforcement learning problem where each CognitiveCache edge considers its caching strategy together with the caching behaviours of its neighbours edges as part of the environment. CognitiveCache relies on actor-critic based

RL approach [5], [57] and utilises Long-Short Term Memory (LSTM) [40]) as the (deep) recurrent neural network architecture for the actor-critic networks to learn a model of the environment. Figure 2 shows an overview of CognitiveCache framework. Note that we will use the term edge, agent and node interchangeably.

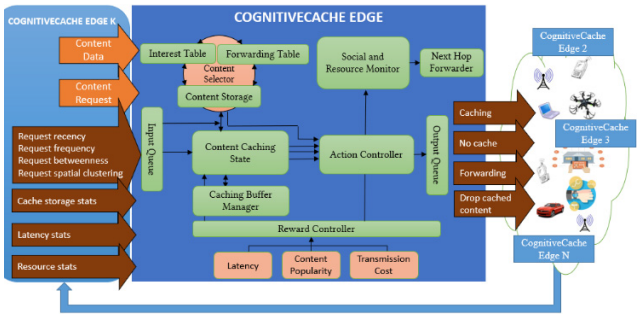


FIGURE 2. CognitiveCache framework.

At every epoch t, CognitiveCache edge adaptively and collaborative captures and predicts the dynamic changing network environment such as cache availability, complex users' content request patterns using predictive analytics and heuristics proposed in [1]–[3]. As a result, CognitiveCache edge can form a state of the network environment observed by not only itself but also its neighbour edges. CognitiveCache relies on actor-critic based RL approach [5], [57]: the actor network controls how the edge behaves by learning the optimal policy, taking the state as input and outputs the best caching decisions such that whether to cache or evict/drop a list of contents; the critic network evaluates and gives feedback to the selected caching decisions to keep improving in real time the caching decision policy. After taking the caching decisions, each CognitiveCache edge receives a reward based on the popularity of cached content, content transmission latency and costs at the next epoch $t+1$. The reward and the next state observed by CognitiveCache at $t+1$ help to keep improving the caching decisions such that to maximise the cumulative reward in order to eventually improve the average cache hit ratio, reduce latency and transmission cost. One disadvantage of standard actor-critic based RL approach is that seeking for the best caching actions is undirected and slow to converge [65]. Thus, our CognitiveCache utilises (deep) recurrent neural network (specifically, LSTM [40]) as the architecture for the actor-critic networks to learn a model of the environment that helps to capture long-term temporal dependencies, predict next observations/states and rewards based on current observations/states and actions in the partially observable environment. At the result, CognitiveCache converges quickly to achieve good caching decision policy.

Figure 3 shows an example of states and caching action transitions of CognitiveCache. CognitiveCache edge maintains and exchange with other neighbour edges the state of its caching storage (i.e. which contents have been cached) and a list of content popularity it observes. The CognitiveCache edge in high cache state means it has not cached much and still has plenty of space in its caching storage. Thus the edge may freely decide to cache high, medium or low popular content. This makes the transitions from high cache to medium and low cache. When the edge is in medium or low cache state, it will be more selective about which contents to cache so that it only caches high (or medium) popular content as it will need to carefully drop its old cached contents (i.e. ones which are expired or have lowest content popularity) to make more caching space. When a CognitiveCache edge does not receive many content request (i.e. low interest), it may decide to not cache the new content and at the same time, the edge will be able to drop its old (or expired) cached contents that increases the cache availability.

Cache \ Interest	High	Medium	Low
High	Yes	Yes	Yes
Medium	Yes	Yes	No
Low	Yes	No	No

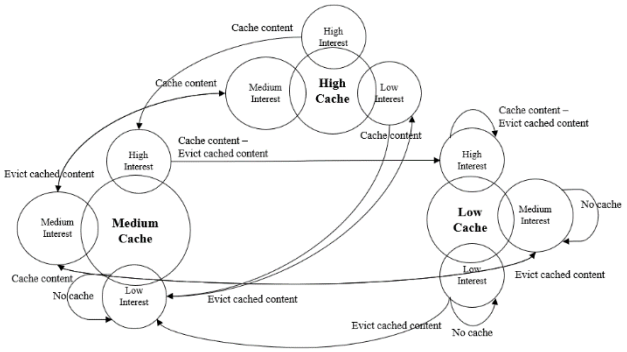


FIGURE 3. Example of CognitiveCache state-action transitions.

We model CognitiveCache system as a network G that consists of a set N of edges n_i ($n_i \in N$) and a local CDN server denoted as C. We define a set of neighbours of each edge n_i as NE_i. We assume that each CognitiveCache edge $n_i \in N$ in the network has a cache of size CS_i. We denote with O a set of content files that can be requested by the network. Each content $o_k \in O$ has the size δ_k. Content o_k consists of an array of chunks $o_{k,l}$. For simplicity, we assume all chunks $o_{k,l}$ of a single content o_k will have the same chunk size $\delta_{k,l}$ without losing generality. We also denote r_k^t as the interest about content o_k at time t. We denote $p_{i,k}^t$ as the popularity of content o_k observed and predicted by the edge n_i during the interval time Δt. Content popularity implies the probability of how likely content will be requested in a period of time. Contents are globally popular if they have been requested by a high number of subscribers coming from different areas in the networks while localised highly popular contents are those which have been requested by subscribers from the same location. Each edge $n_i \in N$ in the network receives certain requests of content o_k at time t, denoted as local content request rate $q_{i,k}^t$. In addition to this,

$z^t_{i,k}$ is denoted as the aggregated request rate of the content o_k observed from all the collaborative neighbours of n_i at time t. We denote $x^t_{i,k} \in \{0, 1\}$ as whether edge n_i has a cache of content o_k at time t or not. We denote $y^t_{i,j,k}$ as whether a content o_k requested within edge n_i area is cached by n_j. $n_j \in \{NE_i, C\}$ is either a neighbour edge n_i or CDN server at time t. Table 1 summarises the main notations used in this paper.

TABLE 1. Table of main notions used in the paper.

Notation	Meaning
N	a set of edges
C	local CDN server
CS_i	cache size/capacity of edge $n_i \in N$
O	a set of content files
$p^t_{i,k}$	the popularity of content o_k observed and predicted by the edge n_i at time t
$x^t_{i,k}$	binary value indicating whether edge n_i has a cache of content o_k at time t.
$y^t_{i,j,k}$	binary value indicating whether a content o_k requested within edge n_i area is cached by n_j at time t. $n_j \in \{NE_i, C\}$ is either a neighbour edge or CDN server.
$u^t_{i,j,k}$	beneficial utility value when caching content o_k at the edge n_j for the edge n_i
$l^t_{j,i}$	latency of transmission between $n_j \in \{NE_i, C\}$ and the edge n_i
$c^t_{j,i}$	transmission cost between $n_j \in \{NE_i, C\}$ and the edge n_i

We measure the caching performance as the product of cache hit ratio, reduction in latency and transmission cost. The objective of our optimization is to compute cache content placement among the edges such that the aggregate benefit is maximized. We formulate the optimal cache content placement problem as follows in equation 1.

$$max : \sum_{o_k \in O} \sum_{n_i \in N} p^t_{i,k}.x^t_{i,k}.u^t_{i,j,k} \tag{1}$$

$$\text{Subject to} : \sum_{o_k \in O} x^t_{i,k} \leq CS_i \tag{2}$$

$$x^t_{i,k} \in \{0, 1\} \tag{3}$$

Equation 2 ensures that each edge does not exceed its caching capacity. Equation 3 restricts the optimization caching decision as a binary value indicating whether to cache content or not. $u^t_{i,j,k}$ is the beneficial utility value when caching content o_k at the edge $n_j \in \{NE_i, C\}$ which is requested from the area of the edge n_i. In the short term, $u^t_{i,j,k}$ is the beneficial utility for caching content o_k at the edge n_j for the edge n_i. Note that $i = $ j implies local edge cache hit. $u^t_{i,j,k}$ is defined as the inverse proportion of the additive value of latency and transmission cost as in equation 4 below:

$$u^t_{i,j,k} = \frac{1}{\alpha l^t_{j,i} + \beta c^t_{j,i}} \tag{4}$$

where $l^t_{j,i}$ and $c^t_{j,i}$ are the latency and transmission cost between $n_j \in \{NE_i, C\}$ and the edge n_i, α and β weights the importance of latency and transmission cost. As discussed above regarding the relation between local hit, neighbour hit and CDN hit, we summarise the relation of $l^t_{j,i}$ and $c^t_{j,i}$ as in equations 5 and 6 below:

$$l^t_{i,i} < l^t_{j \in \{NE_i\},i} < l^t_{j \in \{C\},i} \tag{5}$$

$$c^t_{i,i} < c^t_{j \in \{NE_i\},i} < c^t_{j \in \{C\},i} \tag{6}$$

Our cache content placement optimisation problem is a typical Integer Programming program which is NP-Complete [1], [6]. Achieving a solution working in real time with global optimality is non-trivial [6], [18], [56] regarding the partial network knowledge, the dynamic of mobile users and their content requests. In this paper, we explore how to develop multi-agent DRL caching approach for the mobile edge-cloud scenario where not only individual edge can capture and predict the spatial-temporal locality of content traffic patterns [3], [4] to make data-driven caching solution but also multiple edges can collaborate with each other to solve this optimisation problem.

B. COGNITIVECACHE – A MULTI-AGENT DEEP REINFORCEMENT LEARNING-BASED CONTENT CACHING AT THE EDGES

We propose our novel state, action, reward design followed by the algorithm and architecture of our multi-agent deep reinforcement learning caching framework CognitiveCache.

1) COGNITIVECACHE STATE, ACTION AND REWARD DESIGN

Each CognitiveCache edge in the network maintains a historical record of state-action-reward tuples $\langle s_0, a_0, r_0\rangle$, $\langle s_1, a_1, r_1\rangle$ of its own and its neighbour edges. Our caching's multiagent environment is not globally observable but rather partially observable where each edge is able to capture the environment state and communicate with its neighbours. We describe our novel design of state and action spaces, and the reward function of the CognitiveCache agent as follows:

State space: CognitiveCache maintains the state s^t_i of edge n_i at time t as $s^t_i = \{x^t_i, p^t_i\}$ where $x^t_i = \{x^t_{i,0}, x^t_{i,2}, \ldots, x^t_{i,k}\}$ is the binary value indicating whether edge n_i has a cache of a list of contents $o_k \in O$ at time t and $p^t_i = \{p^t_{i,0}, p^t_{i,2}, \ldots, p^t_{i,k}\}$ is the popularity of a list of content $o_k \in O$ observed and predicted by the edge n_i at time t. To capture and infer the content request demand, simply logging the number of content requests is not sufficient. We utilise the content predictive analytics proposed in [3] to capture the spatial-temporal locality of content requests more accurately and responsively. Specifically, $p^t_{i,k}$ is resolved by the combination of temporal (request frequency, recency, betweenness [3]) and spatial content heuristics [3] in order to allow CognitiveCache to capture and predict the locality trend of content request patterns over time in different locations and avoid losing valuable contents by reducing the caches for one-timers contents [3]. The input state of an edge includes its own

observed state and its neighbours' states, denoted as:

$$s_i^t, \left\{s_j^t\right\} \quad \forall j \in NE_i \tag{7}$$

Action space: At every epoch, after observing the input state of the environment (i.e. cache storage state and content request popularity), CognitiveCache makes the action based on its policy. The action a_i^t of edge n_i at time t is defined as:

$$a_i^t = \left\{a_{i,k}^t, \forall k \in O\right\} \tag{8}$$

in which for every iteration, edge n_i has to decide a list of contents to be cached $a_{i,k}^t = 1$ and a list of content to not be cache $a_{i,k}^t = 0$ (or be removed if the cache storage is full). Each edge decides the best actions based on the input state. CognitiveCache seeks for high entropy in our policy to explicitly encourage exploration that assigns equal probabilities to actions that have the relatively same Q-values. This also avoids CognitiveCache repeatedly selecting a particular caching action that could exploit some inconsistency in the approximated Q function.

Reward space: After taking the caching actions, each CognitiveCache edge receives a reward r_i^t. We define the reward r_i^t of edge n_i at time t after taking a list of actions as:

$$r_i^t = \alpha \sum_{o_k \in O} p_{i,k}^t u_{i,i,k}^t + \beta \sum_{o_k \in O} \sum_{n_j \in NE_i} p_{j,k}^t u_{i,j,k}^t \tag{9}$$

in which $p_{i,k}^t$ is the popularity of contents in the next epoch, $u_{i,j,k}^t$ is beneficial utility value when caching content o_k at the edge n_j for the edge n_i. As shown in Equation *4*, $u_{i,j,k}^t$ is the inverse proportion of additive value of latency and transmission cost. $u_{i,i,k}^t$ means local cache hit. In Equation *9*, we consider the rewards of both local edge and its neighbours when improving the caching policy. This is because the local edge can serve requests from its neighbours and acquire some reward value. In our model, we assume $\alpha > \beta$, i.e the local edge reward has higher weight than the neighbour reward, thus the policy updating is more driven to local cache hit which has lower latency and transmission cost for delivering content to subscribers.

2) COGNITIVECACHE ARCHITECTURAL OVERVIEW AND PSEUDO-CODE

In our context with multiple edges (e.g. BSs, APs, mobile devices, vehicles, etc.), a single centralised learning agent is not applicable as every individual edge should have its own caching policy driven by its observed user content request patterns and the single-central agent could have limited scalability due to the explosive action space of massively distributed edges [18], [56]. Fully-cooperative approaches also suffer from scalability and stability performance [18]. Therefore, we propose CognitiveCache framework based on multi-agent independent DRL [38] where each CognitiveCache edge adaptively considers its own caching strategy while collaborating with its neighbours such that the input states of an edge will involve its own state together with the states of its neighbours. Figure. 4 shows the CognitiveCache multi-agent deep reinforcement learning-based caching framework.

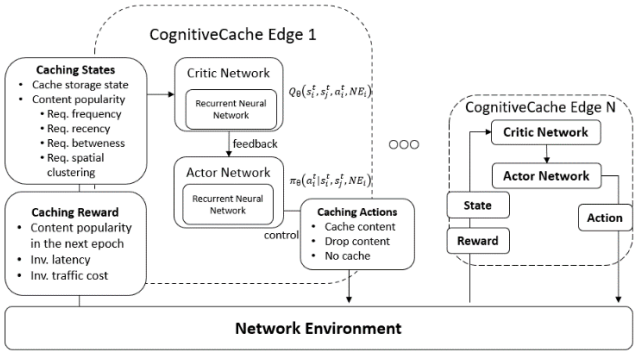


FIGURE 4. **CognitiveCache independent multi-agent DRL-based caching framework.**

We propose to utilise soft actor-critic (SAC) method [5], [57] for the multi-agent RL [18], [24] which optimizes a stochastic policy in an off-policy manner. The actor network controls the caching decisions/actions and the critic network evaluates and gives feedback on the chosen caching decisions to update the caching policy. Soft actor-critic [5], [57] is more sample efficient and more robust to brittleness in convergence compared to other approaches such as [56]. We base our work on [5], [57], which was originally designed for continuous actions splace, to provide an alternative version of the soft actor-critic (SAC) algorithm that is applicable to discrete action settings. In addition to searching for maximum rewards, SAC algorithm maximises the entropy of the policy.

Regarding the architecture of deep neural network for pre-training actor and critic networks, we utilise recurrent neural network (RNN) or more specifically long short term memory (LSTM) [40] as shown in Fig. 5, which is a state-of-the-art learning model that is typically used for time series prediction. This allows CognitiveCache to capture and explore the hidden users' temporal content request patterns as well as address the problem of large input space compared to other traditional deep neural network [56].

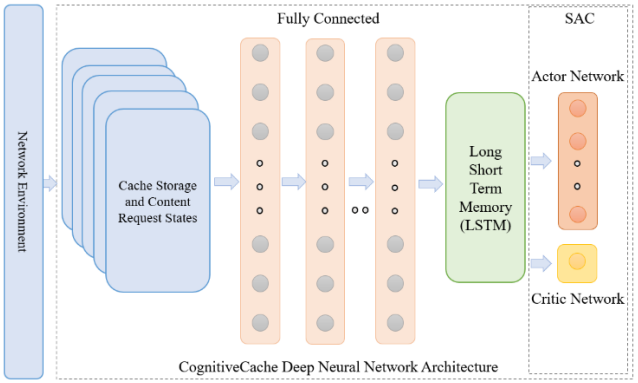


FIGURE 5. **CognitiveCache edge deep learning architecture.**

We describe the SAC objective function consisting of both reward and entropy function [5], [57] in Equation 10 as

follow:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_i^t, a_i^t)\sim\tau_\pi}[\gamma r_i^t + \alpha H\left(\pi \cdot |s_i^t\right))] \quad (10)$$

where π is a policy, $\gamma \in [0, 1]$ is the discount rate, τ_π is the trajectories distribution by policy π, α is the parameter indicating the importance of the entropy term versus the reward [5], [57]. $H\left(\pi \cdot |s_i^t\right))$ is the entropy of the policy π at state s_i^t. $H\left(\pi \cdot |s_i^t\right)) = -log\left(\pi \cdot |s_i^t\right)$.

In actor-critic based caching approach, the actor network controls how the edge behaves by learning the optimal policy, taking the state as input and outputs the best caching decisions. The critic network evaluates the action by computing the value function. CognitiveCache utilises SAC which makes use of three functions: a state value function V, a soft Q-function Q, and a policy function π. We train the three function approximators in line with [5]. The soft state value function for discrete action space [5] is defined as:

$$V\left(s_i^t\right) = \pi(s_i^t)^T[Q\left(s_i^t\right) - \alpha log\left(\pi(s_i^t)\right)] \quad (11)$$

We train the soft Q-function parameterized by θ by minimising the error function [5]:

$$J_Q(\theta) = \mathbb{E}_{(s_i^t, a_i^t)\sim D}[\frac{1}{2}(Q_\theta(s_i^t, a_i^t) - (r_i^t + \gamma \mathbb{E}_{s_i^{t+1}\sim\tau_\pi}[V_{\bar{\theta}}(s_i^{t+1})]))^2] \quad (12)$$

where D is the experience replay buffer [5], [57]. The policy is then updated in a direction that maximises the potential rewards. Finally, we train the policy network π parameterized by ϕ by minimising the error function [5]:

$$J_\pi(\phi) = \mathbb{E}_{s_i^t\sim D}[\pi\left(s_i^t\right)^T[\alpha log\left(\pi_\phi(s_i^t)\right) - Q_\theta(s_i^t)]] \quad (13)$$

We provide CognitiveCache pseudo code in Table 2. CognitiveCache updates all the network functions of every individual edge during each epoch in an experience-replay manner. After the actor-critic based training for our CognitiveCache, the actor network can be utilised to make caching decisions for every single edge. More specifically, CognitiveCache framework consists of two phases: 1) Offline-training: the actor and critic networks are constructed and pre-trained with a sufficient number of historic transition samples in order to achieve good initial parameters for phase 2. 2) Online control: start with a set of parameters bootstrapped in phase 1, in each epoch t, if the requested content is already cached (local cache hit), the edge immediately sends the requested content to the subscribers. If the requested content is not cached, the edge observes state s_i^t of itself and its neighbours resolved based on [3] and obtains the Q-value from the actor-critic networks. Then, a list of action a_i^t are selected based on π-policy, whether to cache the content or evict/drop it. CognitiveCache edge is encouraged to explore different possible actions that assigns equal probabilities to actions that have the same or close Q-values. After the action a_i^t is executed, the edge observes the reward r_i^t and next state s_i^{t+1} on which the action policy keeps updating for the next epoch time t+1.

TABLE 2. CognitiveCache's pseudo code.

Handle content requests arrival

When the *Requests* (or interests) of *Content* are received at a CognitiveCache *Edge*:
if *Edge* has already cached *Content*:
    send *Content* to *Request* 's subscribers
else
    listRequest = {}
    if listRequest [] not contains *Request*
        listRequest.add(*Request*)
    end if
    if listSubscribers[] not contains *Request's* subscriber
        listSubscribers.add(*Request* 's subscriber)
    end if
    for each request **in** listRequest [] **do**:
        if request is expired:
            listRequest.remove(request)
        end if
    end for
    LSTM as function approximator:
        initial policy parameter, Q-function parameters, empty replay buffer
        set target parameters
    repeat
        updateContentStorage()
        updateContentRequest()
        cachingStates = {{x},{p}} // {{low, medium, high cache},{low, medium, high interest}}
        for each Neighbour **in** scan **do**:
            exchangeContentPopularityInfo(Neighbour.Reputation)
            exchangeCacheStateInfo()
            Edge.updateContentPopularity()
            Edge.update(cachingStates)
        end for
        cachingActions = {a} // {cache content, drop old content, no cache}
        select caching action *a* from caching policy
        observe next caching state *s'*, reward *r* and store *(s,a,r,s')* in replay buffer
        while update
            randomly sample a batch of transitions *(s,a,r,s')* from replay buffer
            compute target for Q-functions
            update Q-functions and caching policy by one step of gradient descent
            update target network
        end while
    until convergence

The transition $(s_i^t, a_i^t, r_i^t, s_i^{t+1})$ is stored in CognitiveCache memory at the end of each time period.

IV. EVALUATION

This section provides rich multi criteria evaluation of CognitiveCharge, first describing realistic experimental datasets with a case study, then introducing a set of benchmark and state-of-the-art caching policies as competitive caching algorithms.

We use Foursquare [26] and Twitter [28] datasets as real traces to drive user content requests in two very different network scenarios: New York [26] and London [28]. Foursquare New York dataset [26] is collected through location-based service Foursquare API (https://developer.foursquare.com/) describing the spatial-temporal locality of content requests in terms of user interests at public venues. We assume the contents represent 14,550 requests of 5101 users in different

locations of New York City during the period of one week. Each record is associated with its timestamp, its GPS coordinates and its semantic meaning.

Similarly, the data of Twitter London [28] was collected in one week, containing 15,602 geotagged tweets, posted by 5869 users. Each tweet consists of a timestamp and GPS coordinates (latitude and longitude).

We focus on a particular edge-cloud community of users: students in big universities campuses in New York and London. Students have different heterogeneous mobility patterns, interests and content request usage. This gives us statistically sufficient diversity to evaluate CognitiveCache and competitive caching protocols performance in different contexts. Without loss of generality, we split the chosen area into multiple 1 km $\times$ 1 km small grids and assume a CognitiveCache edge positioned at the centre of each grid will serve the content requests coming from its area. Students' requests may range from texts (e.g. bus schedule information) to pictures and videos. Based on our traces analysis, the requested contents are highly skewed so that a lower number of contents are requested more frequently by the users/subscribers. This is because the students may share some common interests such as bus schedule, travel, nightlife, restaurants, shopping, cinema, etc. Table 5 and 6 show the content topic distribution in New York and London scenarios in which the three highest popular contents in New York belongs to topic of college, coffee shop and subway while that in London are transportation, college and coffee shop. As shown in Figure 5 - 7, the content request has shown a certain level of locality in each area. Moreover, our traces analysis shows certain similarities among neighbouring edges will leverage potential collaborations between the edges.

We design the CognitiveCache learning model using Python and Tensorflow [23], running on a machine with GTX 1050 Ti GPU card, Intel I7 3.6 GHz CPU cards and 16GB memory. As shown in Table 3, we set discount factor as 0.99 and the learning rate for both the actor-critic networks are 1e-4. The hidden layers' size is 256. The number of iterations is 20000. We use 70% data for training and 30% of data for evaluation.

TABLE 3. Values of the learning parameters.

Parameter	Value
Discount factor	0.99
Learning rate (actor network)	1e-4
Learning rate (critic network)	1e-4
# of hidden layers	256
# of iterations	20000
Training:testing	70:30

We perform the evaluation across a range of criteria: local edge (cache) hit ratio, neighbour (cache) hit ratio, latency and transmission cost in the face of vastly different mobility, workloads and content traffic patterns against multiple state-of-the-art and benchmark protocols: LRU [9], LFU [10], ProbCache [60] and DQNCache [55]. Least frequently used (LFU) [10] measures the frequency of content requests at a caching node to make caching decisions. The content receiving more frequent requests will be cached because it will have higher chance of being requested again in the future. Least recently used (LRU) [9] records the time stamp of a content request locally in order to make caching decisions based on how recent the content requests are. ProbCache [60] keeps a copy of content in a cache along a path with a probability which is calculated based on path lengths and multiplexes content flows. DQNCache [55] is a deep reinforcement learning-based caching in hierarchical content delivery networks. The proposed framework DQNCache relies on Deep Q Networks to learn optimal caching policy in an online manner. DQNCache belongs to value-based algorithm which tries to find optimal value function which is a mapping between an action and a value to find better action with higher value.

Local edge (cache) hit ratio is the proportion of requests being served directly from the local edge's cache. Neighbour (cache) hit ratio is the proportion of requests being served indirectly from the neighbour edges; local edge hit ratio and neighbour hit ratio together show the proportion of content requests being satisfied by the edges instead of the CDN server. Latency means end-to-end average latency of all content requests in a time period. Note that we assume the transmission latency between any edge to the CDN is 3 times of the latency between any two neighbouring edges. Transmission cost is the total traffic cost when forwarding requests/receiving contents to/from neighbour edges or CDN server. We describe our experiments with increasing cache capacity which is the storage capacity constraint implying the maximum number of contents that an edge can cache. Smaller cache capacity size offers more selective cached contents, thus requires more accurate and more robust caching algorithms. All experiments are repeated ten times and averaged. The detailed simulation parameters are shown in Table 4.

TABLE 4. Values of the simulation parameters.

Parameter	Value
Complex temporal network topologies	New York, London
Content request pattern	Foursquare, Twitter
Number of nodes	5101 – 5869
Simulation duration	7 days
Number of contents	14550 – 15602
Request rate	1-25 request/min
File size	1 MB - 8.4 MB [27]
Interest packet size	8 kB - 128 kB [27]
$\frac{\text{Cache size}}{\text{Total content population}}$	1% -6%

Figure 6 shows the overall temporal trends or patterns of users' content traffic during weekdays and weekend: if content is requested at a certain interval of time, it is highly likely it will be requested again in near future. Contents are

TABLE 5. Content topic distribution in New York scenario.

Topic	Number of contents
College	4033
Coffee shop	3432
Subway	2719
Park	1930
Restaurant	1206
Others	1230

TABLE 6. Content topic distribution in London scenario.

Topic	Number of contents
Transportation	5890
College	2922
Coffee shop	2441
Subway	1890
Park	1010
Others	1449

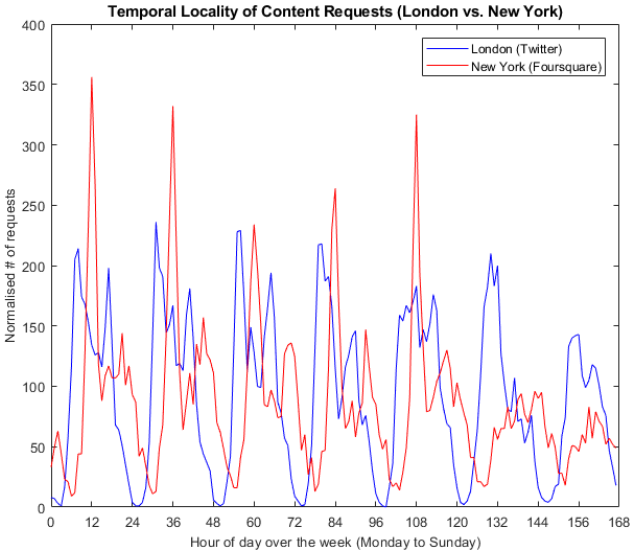
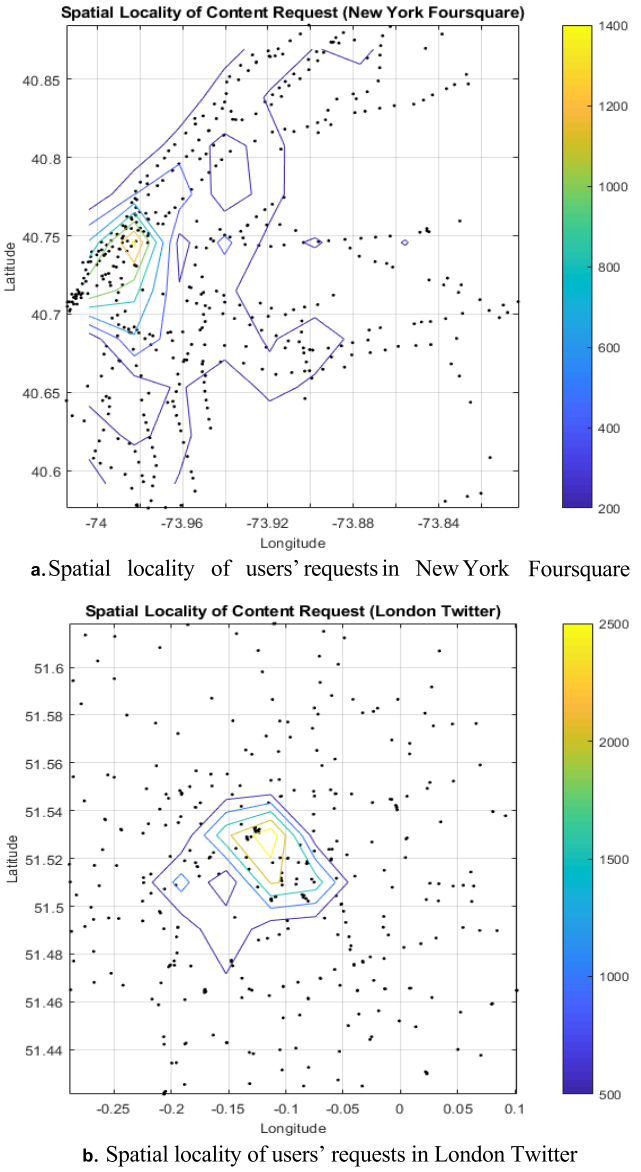


FIGURE 6. Temporal locality of users' requests in New York (Foursquare) and London (Twitter).



a. Spatial locality of users' requests in New York Foursquare

b. Spatial locality of users' requests in London Twitter

FIGURE 7. (a) Spatial locality of users' requests in New York Foursquare. (b) Spatial locality of users' requests in London twitter.

not requested randomly and independently over time but at a certain time interval, before its popularity gradually fades out. Foursquare New York experiences surge of traffic at peak hours during weekdays and low average number of requests during weekend. On the opposite, Twitter London has a more uniform distribution of content traffics, although it still experiences surge of traffic twice a day, during a week.

Figure 7a and Figure 7b show the user content request distribution in New York Foursquare [26] and London Twitter [28]. We show that the locations of mobile subscribers imply different degrees of similarity in content request. Subscribers within a community or from two relatively close clusters with each other are more likely to have similar content request patterns compared to those from long-distance subscribers or regions far apart. This captures the interplay between geographical diversity of the users and their content request patterns. New York Foursquare has high degree of power-law distribution [13] such that there is a small important number of nodes which are highly connected and there's a trailing tail of nodes with a very few connections [13]. In London Twitter, although the degrees of its nodes still follows power-law model, its topologies have more uniform connectivity distribution compared to New York Foursquare such that popular nodes may become extremely less popular, and emerging new nodes may become extremely high popular in a very short time. London Twitter has shorter average paths and lower clustering compared to New York Foursquare. It also has lower publisher-subscriber density compared to New York Foursquare (35.1 users/km^2 and 64.8 users/km^2 respectively).

We investigate the influence of edge caching storage capacity on a range of metrics. We vary the edge caching

storage capacity from 200 contents which is equivalent to 1% of the total content population to 1200 contents (which is equivalent to 6% of the total content population). Figure 8 and Figure 9 show the cache hit ratio of high popular contents in local edge and in neighbour edges for both New York and London scenarios. We show that CognitiveCache outperforms all other competitive approaches, improving more than 52% cache hit ratio for New York Foursquare and 88% for London Twitter. CognitiveCache has good performance in both New York Foursquare with high-degree scale-free network community where a small number of contents are requested a lot and in the London Twitter scenarios where the structure of community and user requests changes significantly over time. Higher cache space leads to the bigger gap between CognitiveCache and others. CognitiveCache allows 91% of high popular content to be cached and served to subscribers within the local edge. When the local cache space is relatively small, high popular contents which are not be cached locally will be redirected and served via the neighbour edges rather than sending to the CDN server. This is due to CognitiveCache is able to collaboratively learn the time-series and spatial content request patterns from the historical observations while taking into account the states of itself and its neighbours' edges. The neighbour hit ratio of CognitiveCache decreases when the capacity increases. This is because most highly popular requests will be served locally when cache space is increased. At the result, the percentage of neighbour cache hit is expected to reduce. CognitiveCache outperforms single-agent DQNCache [55] in both New York and London scenarios as DQNCache [55] only aims to maximise the individual performance for each individual edge without considering the state of its neighbours. Probability-based caching algorithm ProbCache [60] has a better performance compared to benchmarking LRU and LFU caching algorithms.

Figure 10 and Figure 11 show the cache hit ratio in local edge and in neighbour edges for low popular contents in New York and London scenarios. We show that while CognitiveCache allows a majority of high popular contents to be served by local edge, it also enables the cache hit ratio of low popular contents to be 80% in both New York Foursquare and London Twitter while that of DQNCache, ProbCache, LRU and LFU are 51%, 33%, 27% and 21% respectively. CognitiveCache preserves most of its cache space for predicted high popular contents while still being able to serve 39% of low popular requests locally and additional 41% via its collaborative neighbour edges' cache. When the cache space is getting larger, each CognitiveCache considers not only its local hit but also offers a proportion of its cache space to serve its neighbours. DQNCache, ProbCache, LRU and LFU has poor caching performance for low popular content, especially in London scenario with very high dynamic mobile subscribers and users requests.

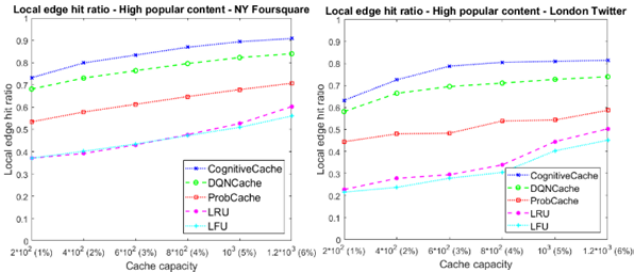


FIGURE 8. Local edge hit ratio vs. Cache capacity for high popular content in NY and London.

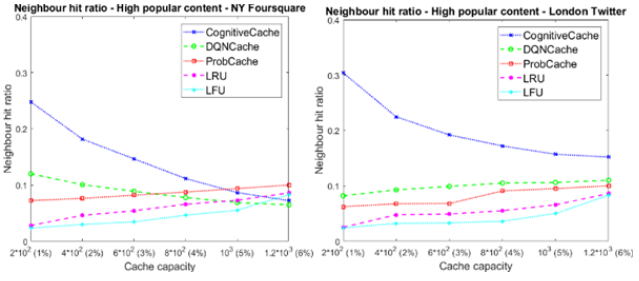


FIGURE 9. Neighbour hit ratio vs. Cache capacity for high popular content in NY and London.

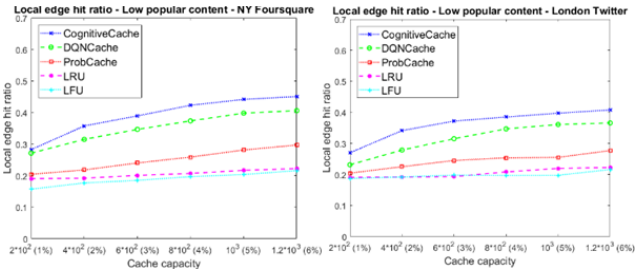


FIGURE 10. Local edge hit ratio vs. Cache capacity for low popular content in NY and London.

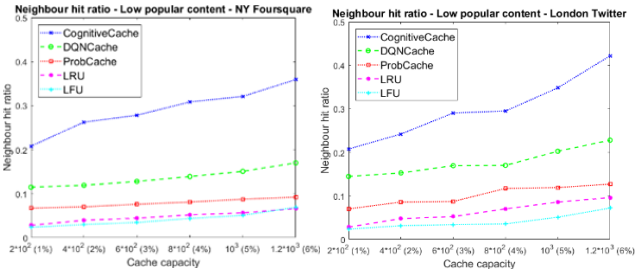


FIGURE 11. Neighbour hit ratio vs. Cache capacity for low popular content in NY and London.

Figure 12 and Figure 13 show the average latency in local edge and in neighbour edges for low popular contents in New York and London scenarios. CognitiveCache can reduce 33%, 47%, 66%, 71% latency compared with DQNCache, ProbCache, LRU and LFU respectively. In Figure 8-11, we show that CognitiveCache can successfully serve 91% of high popular contents within 9ms and 80% of low popular contents within 19ms for New York scenario while that of London is 15.7ms and 19.4ms respectively. CognitiveCache has better performance in delay compared to competitive caching protocols since it benefits from its well-identified

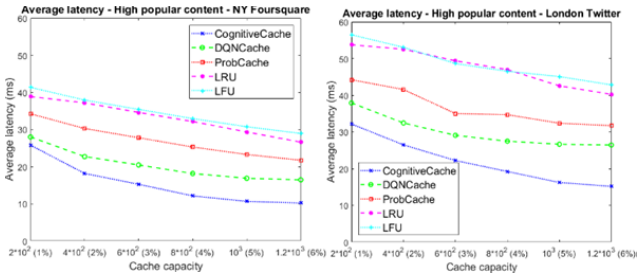


FIGURE 12. Average latency vs. Cache capacity for high popular content in NY and London.

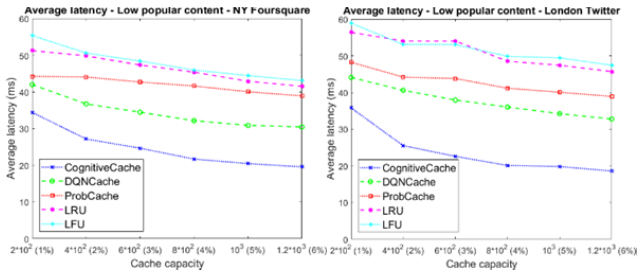


FIGURE 13. Average latency vs. Cache capacity for low popular content in NY and London.

states consisting of its predictive request demand and the caching state that allow it to learn and adapt faster and more accurate to the dynamically changing of mobile subscribers and their requests.

Figure 14 shows the average transmission cost of sending requests and receiving contents either from local edge, neighbour edges or from the CDN server in New York and London scenarios. Note that we assume if a content request achieves local hit, the transmission cost is relatively lower compared to neighbour hit. In turn, the transmission cost of neighbour hit is lower compared to that of the CDN server.

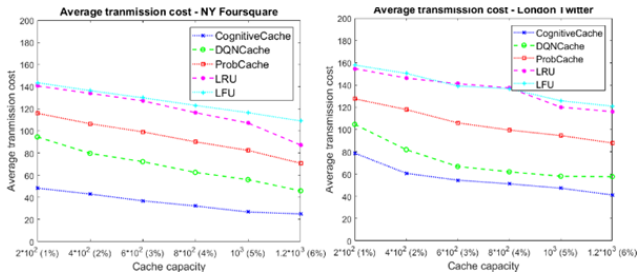


FIGURE 14. Average transmission cost vs. Cache capacity in NY Foursquare and London twitter.

We show that CognitiveCache reduces 23%, 75%, 83%, 87% transmission cost compared with DQNCache, ProbCache, LRU and LFU respectively. This is due to CognitiveCache is able to serve a majority of high popular content locally while allowing low popular content to be cached in neighbour edges, thus minimizing the number of requests being sending to CDN server. Since obtaining contents, especially video from a local edge or even neighbouring edge is quicker and more cost-beneficial compared to that from the CDN server, CDN content retrieval should be the lowest priority.

Figure 15 shows how our CognitiveCache caching captures, predicts and responds to the user requests in real time in two very different scenarios: New York Foursquare and London Twitter. The historical content request patterns offer valuable resources for our data-driven caching solution as we show that CognitiveCache can capture and predict the temporal-spatial locality of users' content requests that are leveraged for highly accurate, responsive and cost-effective CognitiveCache caching decisions. CognitiveCache enables more responsiveness to the rising trend of newly high popular contents and fading out of older contents over time as well as avoid one-timer contents [47] and mitigate flash crowd effect [3].

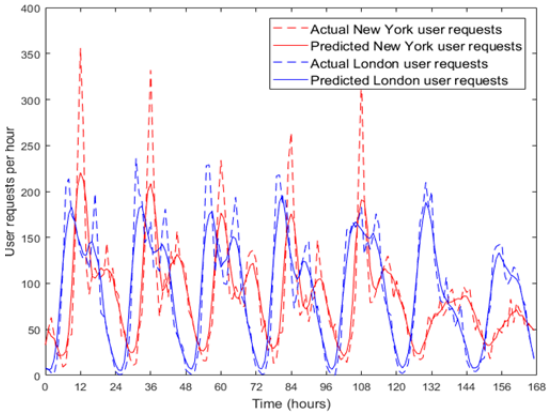


FIGURE 15. Actual and CognitiveCache predicted users request in New York Foursquare and London Twitter.

V. CONCLUSION

In this paper, we proposed CognitiveCache caching framework based on multi-agent deep reinforcement learning which can tackle the complex challenges of bringing contents as close as possible to the mobile users and improve the quality of service in mobile edge-cloud networks. We design a novel space of states, actions to leverage the temporal-spatial locality of content requests that enables more accurate and responsive caching decision making. We evaluate our CognitiveCache proposal against benchmark and competitive caching models: DQNCache [55], ProbCache [60], LRU [9] and LFU [10] over two very different real-world network topologies: New York [26] and London [28]. We show that our caching framework consistently outperforms the benchmarking and state-of-the-art algorithms, increases the cache hit ratio while minimising the latency and transmission costs.

In future work, we plan to explore and propose a novel incentive mechanism that incentivise all edges in the network to collaborate and share their caching space with fairness [6] and security concern, avoid the selfish and malicious behaviours of users in real world. In addition to this, we will investigate new privacy-aware and energy-aware CognitiveCache by building on and extending works in [32], [44]

for edge privacy awareness and [36], [50] for edge energy efficiency. In addition, while complex ML/RL techniques and algorithms help to analyse a huge amount of historical data to gain deeper insight of network environments, predictive analytic and heuristic-based approaches [1]–[3], [29] allow predictive adaptive response to changing local conditions in real time and at low cost. This opens up opportunities for future work to innovate and redefine the ML/RL based caching algorithm assisted with real-time predictive analytics and heuristics to improve the accuracy, scalability and efficiency for content services in mobile heterogeneous networks.

REFERENCES

[1] M. Radenkovic and V. S. H. Huynh, "Collaborative cognitive content dissemination and Query in heterogeneous mobile opportunistic networks," in *Proc. 3rd Workshop Exper. Des. Implement. Smart Objects*, 2017, pp. 7–12.

[2] M. Radenkovic, V. S. H. Huynh, and P. Manzoni, "Adaptive real-time predictive collaborative content discovery and retrieval in mobile disconnection prone networks," *IEEE Access*, vol. 6, pp. 32188–32206, 2018.

[3] V. Huynh and M. Radenkovic, "Interdependent multi-layer spatial temporal-based caching in heterogeneous mobile edge and fog networks," in *Proc. 9th Int. Conf. Pervas. Embedded Comput. Commun. Syst.*, 2019, pp. 34–45.

[4] A. Dabirmoghaddam, M. M. Barijough, and J. J. Garcia-Luna-Aceves, "Understanding optimal caching and opportunistic caching at 'the edge' of information-centric networks," in *Proc. 1st Int. Conf. Inf.-Centric Netw.*, 2014, pp. 47–56.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*. [Online]. Available: http://arxiv.org/abs/1801.01290

[6] L. Wang, G. Tyson, J. Kangasharju, and J. Crowcroft, "Milking the cache cow with fairness in mind," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2686–2700, Oct. 2017.

[7] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[8] T. Le, Y. Lu, and M. Gerla, "Social caching and content retrieval in disruption tolerant networks (DTNs)," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 1–5.

[9] M. Bilal and S.-G. Kang, "Time aware least recent used (TLRU) cache management policy in ICN," in *Proc. 16th Int. Conf. Adv. Commun. Technol.*, Feb. 2014, pp. 528–532.

[10] G. Zhang, "An optimal cache placement strategy based on content popularity in content centric network," *J. Inf. Comput. Sci.*, vol. 11, no. 8, pp. 2759–2769, May 2014.

[11] T. Wang, P. Hui, S. Kulkarni, and P. Cuff, "Cooperative caching based on file popularity ranking in delay tolerant networks," 2014, *arXiv:1409.7047*. [Online]. Available: http://arxiv.org/abs/1409.7047

[12] D. Mardham, S. Madria, J. Milligan, and M. Linderman, "Opportunistic distributed caching for mission-oriented delay-tolerant networks," in *Proc. 14th Annu. Conf. Wireless On-Demand Netw. Syst. Services (WONS)*, Feb. 2018, pp. 17–24.

[13] S. Bornholdt and H. G. Schuster, *Handbook of Graphs and Networks: From the Genome to the Internet*. Hoboken, NJ, USA: Wiley, 2006.

[14] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proc. 1st ACM/IEEE Int. workshop Mobility Evolving Internet Archit.*, 2007, pp. 1–8.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Cham, Switzerland: Springer, 2006.

[16] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.

[17] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[18] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[19] C. H. Liu, J. Xu, J. Tang, and J. Crowcroft, "Social-aware sequential modeling of user interests: A deep learning approach," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 11, pp. 2200–2212, Nov. 2019.

[20] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[21] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[22] Z. Ning, P. Dong, J. J. P. C. Rodrigues, F. Xia, and X. Wang, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.

[24] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *J. Mach. Learn. Res.*, vol. 7, pp. 1789–1828, Sep. 2006.

[25] (2020). *AWS Cloudfront*. [Online]. Available: https://aws.amazon.com/cloudfront/

[26] (2020). *Foursquare Developer*. [Online]. Available: https://developer.foursquare.com/

[27] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 1–14.

[28] (2011). *Followthehashtag*. [Online]. Available: http://followthehashtag.com/datasets/170000-uk-geolocated-tweets-free-twitter-dataset/

[29] M. Radenkovic and A. Grundy, "Efficient and adaptive congestion control for heterogeneous delay-tolerant networks," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1322–1345, 2012.

[30] A. Said, S. Shah, H. Farooq, A. Mian, A. Imran, and J. Crowcroft, "Proactive caching at the edge leveraging influential user detection in cellular D2D networks," *Future Internet*, vol. 10, no. 10, p. 93, Sep. 2018.

[31] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora, "Graph metrics for temporal networks," in *Temporal Networks* (Understanding Complex Systems), P. Holme and J. Saramäki, Eds. Berlin, Germany: Springer, 2013, pp. 15–40.

[32] M. Radenkovic, "Cognitive privacy for personal clouds," *Mobile Inf. Syst.*, vol. 2016, pp. 1–17, Dec. 2016.

[33] M. Radenkovic and V. S. H. Huynh, "Low-cost mobile personal clouds," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Paphos, Cyprus, 2016, pp. 836–841.

[34] M. Radenkovic and N. Milic-Frayling, "RasPiPCloud: A light-weight mobile personal cloud," in *Proc. 10th ACM CHANT*, 2015, pp. 57–58.

[35] E. J. Oughton, K. Katsaros, F. Entezami, D. Kaleshi, and J. Crowcroft, "An open-source techno-economic assessment framework for 5G deployment," *IEEE Access*, vol. 7, pp. 155930–155940, 2019.

[36] M. Radenkovic and A. Walker, "CognitiveCharge: Disconnection tolerant adaptive collaborative and predictive vehicular charging," in *Proc. 4th ACM MobiHoc Smart Objects Workshop*, 2018, pp. 1–9.

[37] A. P. Silva, K. Obraczka, S. Burleigh, J. M. S. Nogueira, and C. M. Hirata, "A congestion control framework for delay- and disruption tolerant networks," *Ad Hoc Netw.*, vol. 91, Aug. 2019, Art. no. 101880.

[38] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th ICML*, 1993, pp. 330–337.

[39] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, Jun. 2005.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[41] H. Flores, R. Sharma, D. Ferreira, V. Kostakos, J. Manner, S. Tarkoma, P. Hui, and Y. Li, "Social-aware hybrid mobile offloading," *Pervas. Mobile Comput.*, vol. 36, pp. 25–43, Apr. 2017.

[42] A. Said, R. A. Abbasi, O. Maqbool, A. Daud, and N. R. Aljohani, "CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks," *Appl. Soft Comput.*, vol. 63, pp. 59–70, Feb. 2018.

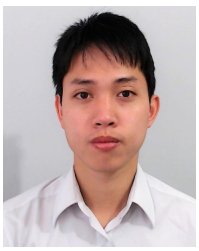
[43] M. Radenkovic, J. Crowcroft, and M. H. Rehmani, "Towards low cost prototyping of mobile opportunistic disconnection tolerant networks and systems," *IEEE Access*, vol. 4, pp. 5309–5321, 2016.

[44] M. Radenkovic, A. Benslimane, and D. McAuley, "Reputation aware obfuscation for mobile opportunistic networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 230–240, Jan. 2015.

[45] E. Lima, A. Aguiar, P. Carvalho, and A. C. Viana, "Impacts of human mobility in mobile data offloading," in *Proc. 13th Workshop Challenged Netw.*, 2018, pp. 1–5.

[46] M. Ruan, X. Chen, and H. Zhou, "Centrality prediction based on K-order Markov chain in mobile social networks," *Peer-Peer Netw. Appl.*, vol. 12, pp. 1662–1672, 2019.
[47] A. Mahanti, D. Eager, and C. Williamson, "Temporal locality and its impact on Web proxy cache performance," *Perform. Eval.*, vol. 42, nos. 2–3, pp. 187–203, Sep. 2000.
[48] M. Selimi, L. Cerdá-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, and J. Crowcroft, "A lightweight service placement approach for community network micro-clouds," *J. Grid Comput.*, vol. 17, no. 1, pp. 169–189, Mar. 2019.
[49] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[50] M. Radenkovic and V. S. Ha Huynh, "Energy-aware opportunistic charging and energy distribution for sustainable vehicular edge and fog networks," in *Proc. 5th Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Paris, France, Apr. 2020, pp. 1–6.
[51] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "DRN: A deep reinforcement learning framework for news recommendation," in *Proc. World Wide Web*, 2018, pp. 167–176.
[52] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 128–135, 2010.
[53] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," *Trends Cognit. Sci.*, vol. 23, no. 5, pp. 408–422, May 2019.
[54] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2018, pp. 1–6.
[55] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 4, pp. 1024–1033, Dec. 2019.
[56] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1–5.
[57] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv:1910.07207*. [Online]. Available: http://arxiv.org/abs/1910.07207
[58] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971
[59] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256. 1992.
[60] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd Ed. ICN workshop Inf.-centric Netw.*, 2012, pp. 55–60.
[61] (2020). *Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/en-gb/services/cdn/
[62] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Commun.*, vol. 22, no. 5, pp. 105–111, Oct. 2015.
[63] D. A. Farber, R. E. Greer, A. D. Swart, and J. A. Balter, "Internet content delivery network," U.S. Patent 6 654 807, May 5, 2003.
[64] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, Oct. 2001.
[65] Bakker, B, "Reinforcement learning by backpropagation through an LSTM model/critic," in *Proc. IEEE ADPRL*, May 2007, pp. 127–134.
[66] M. Radenkovic, V. S. Ha Huynh, R. John, and P. Manzoni, "Enabling real-time communications and services in heterogeneous networks of drones and vehicles," in *Proc. Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2019, pp. 1–5.
[67] J. Zhao, R. Mortier, J. Crowcroft, and L. Wang, "Privacy-preserving machine learning based data analytics on edge devices," in *Proc. AAAI/ACM Conf.*, Dec. 2018, pp. 341–346.
[68] C. H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. O. Wu, and K. Leung, "Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning," *IEEE Trans. Mobile Comput.*, early access, Aug. 30, 2020, doi: 10.1109/TMC.2019.2938509.

MILENA RADENKOVIC received the Dipl.Ing. degree in electric and electronic engineering from the University of Nis, Serbia, and the Ph.D. degree in computer science from the University of Nottingham, U.K. She has authored over 80 papers in premium conference and journal venues. Her research interests span areas of intelligent mobile and disconnection tolerant networking, complex temporal graphs, self-organized security, and distributed predictive analytics, with applications to autonomous vehicles, mobile social networks, smart manufacturing, and predictive telemetry. She was a recipient of multiple EPSRC and EU grants for her research. She has organized and chaired multiple ACM and IEEE conferences and served on many program committees. She is an Editor of premium journals such as the *Ad Hoc Networks* (Elsevier), the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, and *ACM Multimedia*.



VU SAN HA HUYNH received the B.Sc. degree in computer science from the University of Nottingham, in 2016. He is currently a Researcher with the School of Computer Science, University of Nottingham. He has authored in venues, including IEEE ACCESS, ACM MobiCom, CLOSER, PECCS, IEEE WiMob, IEEE FMEC, and IEEE IWCMC. His research interests include opportunistic, self-organizing, distributed, mobile networking and systems, content caching services, mobile edge cloud and fog networks, the Internet of Things, and network science.