Modified reinforcement learning based- caching system for mobile edge computing

Sarra Mehamel, Samia Bouzefrane, Soumya Banerjee, Mehammed Daoui, Valentina Balas

Modified Reinforcement Learning based-Caching system for Mobile Edge Computing

Sarra Mehamel[1,2], Samia Bouzefrane[2], Soumya Banarjee[2], Mehammed Daoui[1], and Valentina E. Balas[3]

[1]University Mouloud Mammeri of Tizi-Ouzou, Algeria
[2]Conservatoire National des Arts et Métiers, Paris, France
[3]Aurel Vlaicu University of Arad, Romania

February 9, 2020

1 Abstract

Caching contents at the edge of mobile networks is an efficient mechanism that can alleviate the backhaul links load and reduce the transmission delay. For this purpose, choosing an adequate caching strategy becomes an important issue. Recently, the tremendous growth of *Mobile Edge Computing* (MEC) empowers the edge network nodes with more computation capabilities and storage capabilities, allowing the execution of resource-intensive tasks within the mobile network edges such as running artificial intelligence (AI) algorithms. Exploiting users context information intelligently makes it possible to design an intelligent context-aware mobile edge caching. To maximize the caching performance, the suitable methodology is to consider both context awareness and intelligence so that the caching strategy is aware of the environment while caching the appropriate content by making the right decision. Inspired by the success of *reinforcement learning* (RL) that uses agents to deal with decision making problems, we present a *modified reinforcement learning* (mRL) to cache contents in the network edges. Our proposed solution aims to maximize the cache hit rate and requires a multi awareness of the influencing factors on cache performance. The modified RL differs from other RL algorithms in the learning rate that uses the method of *stochastic gradient decent* (SGD) beside taking advantage of learning using the optimal caching decision obtained from fuzzy rules.
Index Terms — Caching, Reinforcement Learning, Fuzzy Logic, Mobile Edge Computing.

2 Introduction

In the era of data flood, storing popular contents at the edge is a promising technique to meet user's demands while alleviating the overcrowding on the backhaul. The idea that characterizes Mobile Edge Computing (MEC) is to push the computation/storage resources to the network edges that are in the vicinity of the user equipment (UE) [1]. In the other hand, the introduction of MEC paradigm and the development of advanced applications have led to data-traffic growth as MEC services are relying on big data with different types and of significant amounts. Specifically, *mobile edge computing servers* (MECS) are equipped with computation, analytic and storage capabilities which deal with the significant amount of content requests leading to improve the quality-of-service (QoS) of the network and the quality-of-experience (QoE) to the end user. Local caches are one of the MEC equipment that helps to reduce the request traffic and to improve the response time. In particular, because data requests can be processed by MEC nodes that are close to the user, edge caching will reduce network traffic and speed up the data retrieval. This allows edge caching to relax the need of continuous connectivity by decoupling the producer and the receiver. However, edge caching has been widely investigated due to the requirements in term of information freshness. Caching is considered

a critical bottleneck for the development of MEC systems, since edge caches are located at the edge of the networks and physically closer to the end user. With limited storage, edge caches use different strategies of placement and replacement, which are in general designed to allow spreading only popular contents around the network. In addition, MEC paradigm presents an exclusive opportunity to implement edge caching and to design caching placement and replacement strategies. Therefore, caching popular contents at MECS is considered as a cost effective solution [2] due to the benefits of avoiding network congestion and alleviating the backhaul links. Fulfilling the requirement of both network and the end user is a key point of MEC that consists in a dynamic computational offloading that might cause congestion in the network; which decreases QoS and raises the decision making and the optimization problem on the whole system. Our problem statement is how to jointly encompass the problem of stochastic dynamics of the network and enhance the decision of caching. Several pioneer works about caching in mobile edge computing have been proposed and achieved with quite a good results but these approaches suffer from the following issues:

- Not enough Inputs: They consider only popularity as a key information factor while there are several factors that can effect on the caching decision.
- Strategy combination: They consider either placement or replacement strategies and not the whole caching system.
- Dynamic conditions: Dynamics of the environment and the caching system are not well addressed.
- Tentative isolation: Most of caching strategies do not consider the long term effect of the current caching decision.

Other proposed solutions are optimal but they suffer from a lack of intelligence. In order to fill this gap, We focus in this paper on the design of a novel intelligent caching strategy that tailors mobile edge computing servers among learning methods. As one of learning methods, we propose the use of reinforcement learning (RL) that enables agents to deal with decision making problems by learning through interactions with the environment [3]. Reinforcement learning model has some limitations notwithstanding its efficacy in many fields since scaling is a big challenge [4], [5], [6]. In fact, RL model does not cope well with areas where there is a large space of possible actions or environment states. Hence, many RL models have been used for simplified learning cases [7].

In this context, we define a modified reinforcement learning (mRL) based caching system for MEC, which considers the content features namely frequency, cost and size as well as the device feature like mobility and storage capability. This aims to solve the problem of caching decision making in a realistic way. The main contribution of this paper is to bring a solution to the caching problems in a scenario where caching units are stochastically distributed through MEC servers with a limited backhaul and storage capacity. In particular, we build on a caching system model and define its performance metrics (cache hit ratio and stability). In addition, we define cache storage size, user mobility and content popularity distribution. By coupling the caching decision problem with reinforcement learning layer while relying on recent results from [19], we show that a certain hit ratio can be achieved by increasing the total storage size while the number of MECs is fixed. To the best of our knowledge, our work differs from the previous works in terms of studying a new system model with both placement and replacement aspects of edge caching. The remaining content of this paper is organized as follows: background and review are presented in section 2. The modified RL algorithm and its solution process are given in Section 3. Problem formulation and model development are provided in Section 4. Evaluation and discussion of results are given in section 5. Finally, concluding remarks including future directions are given in Section 6.

3 Background and related work

3.1 Caching in MEC systems

Mobile Edge Computing (MEC) is a decentralized computing concept in which computing resources and application services can be distributed along the communication path from the point storing data to Base Stations (BSs) in wireless networks [53]

Mobile-edge Computing has been also mentioned in ETSI (European Telecommunications Standards Institute) white paper as an environment that offers applications and content providers cloud-computing capabilities at the edge of the mobile network. This environment is characterized by high bandwidth, low latency and a real-time access to radio network information [10]. In particular, generic-computing platforms are used by mobile edge computing servers to be implemented directly at the base stations, which enables the deployment of caching and context-aware services in the vicinity of the mobile users [11]. Consequently, MECS became a unique opportunity to implement and perform edge caching. To describe the scenario of mobile edge caching, we first should discuss the questions of where to cache, what to cache (popular content) and how to cache (caching policies).

Caching improves data availability during disconnection because even when a connection is unavailable, data can still be accessed through the on-board cache, thus allowing mobile clients to continue operating during disconnection [42].

In general, edge caching means that popular contents can be cached in edge nodes such as macro base stations, small base stations or even user equipment. As shown in figure 1, caching a content which is close to the end users can effectively reduce the redundant data traffic and greatly improve the QoE of users [8].

Concerning what to cache, the range of contents is widely increasing including videos, audio files and Internet-of-Things data, which makes the number of reachable contents over the internet extremely huge. Not all of the available contents should be cached, regarding to the fact of limited storage space of edge nodes. In literature, to decide what to cache, content's popularity is one of the prime concern. It represents the probability of requesting contents by users or the frequency of demanding the contents over specific time period. This feature should be taken into consideration as a main factor. Most of the current works that deal with mobile edge caching consider that content popularity follows a static Zipf distribution [9]. But since the user preferences change with time and the user groups associated with edge nodes are varied, considering only content popularity is not enough as a parameter to infer the caching decision. Appropriate contents are selected by edge caches to be stored using caching policies. The caching policies decide to obtain different objectives such as traffic offloading, quality of experience (QoE), energy consumption and so on in order to maximize the cache hit ratio. Caching policies can be divided into categories depending on their characteristics:

- Depending on cache coordination:

Coordinated: caches must exchange information to be efficient, that is, to have a good estimation of where to place the content and to avoid caching the same content in too many caches. Coordinating cache is represented by a distributed session feature that allows multiple instances of a session to broadcast content changes among each other so that each cache is updated.
Uncoordinated: in uncoordinated scheme, each cache is working in an individual way.

- Depending on cache size:

Homogeneous: all caches in base stations have the same size.
Heterogeneous: each cache has a different size.

- Depending on the cooperation between caches

Cooperative: caches cooperate with each other by establishing a cache state that allows other caches to know the different states like in [43].
Non-cooperative: caching decisions are made independently with no need to advertise the information of cache state.

- Depending on where the content is cached:

On path: caching only the contents caught along the downloading path.
Off path: caching the content caught outside the downloading path.

In general, conventional caching policies are divided into two main phases: placement phase and replacement phase:

- The placement phase which is the process that decides whether we should cache and how and when we can cache the contents.

- The replacement phase which is the process that decides which data to drop if there is no free storage space.

To make best usage of edge caching, researchers have prosperously used classical web caching algorithms either for placement or replacement phases relying on popularity as a main factor. For placement, they adapt: *leave copy everywhere* (LCE) which is the default caching mechanism in most caches designs used to minimize the upstream bandwidth demand and downstream latency [12]. In LCE, the popular contents are cached at every cache along the path and it is considered as a homogeneous and non-cooperative caching mechanism. The methods *leave copy down* (LCD) and *move copy down* (MCD) realize the heterogeneous caching in a cooperative way in order to reduce redundancy [12] [13]. Probabilistic mechanisms have been widely used, like the policy referred to *Prob(p)*. It was used as a benchmark scheme in the literature [32] [33] [34]. This mechanism sets to every content a probability p in advance and does not cache the content for a probability equal to *(1-p)*. It is a heterogeneous and non-cooperative strategy. We can also mention *Prob-cache* Prob-cache is another solution identified as an on-path, heterogeneous and cooperative probabilistic caching mechanism. It adds additional fields to each request and content Time Since Inception (TSI) and Time Since Birth (TSB) [14] making the cached content visible to neighboring caches, which enhances the cooperation. Jason Min et al. proposed an explicit cache cooperation named *intra-domain cache cooperation*. Specifically, it maintains two cache summary tables that record the information of the contents currently cached by the content router, and exchange it with other caches [15].

For edge caching, classical web caching was widely used . However, several works have proposed new placement strategies for the edge, such as in [29], where they present the edge buffering as a caching and pre-fetching strategy, pointing out the insufficiency of strategies that rely on past history of the device. Instead, they propose a prediction model based on the aggregated network-level statistics. Unlike blind popularity decisions, authors in [30] proposed a mobility-aware probabilistic (MAP) placing scheme which caches contents at edge servers where the vehicles are connected considering the vehicular trajectory predictions and the time required to serve a content. S. Zhang et al. [31] propose a cooperative edge caching in large scale, where the optimization is carried on the content placement and the cluster size using different features such as traffic distribution, channel quality, stochastic information of network topology and file popularity. Table 1 summarizes some of classical placement strategies.

The replacement phase is the phase that decides which content to evict from the cache. It is categorized into: recency, frequency based such as the *least frequently used* (LFU) [16] and the *least recently used* (LRU) [17], and semantic like *First In First Out* (FIFO) that always replaces the oldest contents in the cache [18]. Recently, an age-based caching replacement was developed in [21]. The previous caching mechanism has been successfully adopted in classical web caching, but it may not meet the performance in the mobile edge caching due to the ignorance of other features and by the use only of popularity. Thus, it would be unable to take advantage of specific characteristics of both the environments and the end user like mobile network topology uncertainty, user mobility, limited storage and cost. In order to overcome these limitations and cope with the future internet usage, the works described in [19] and [20] adjust the varied content properties and its influencing factors using fuzzy control caching system (placement and replacement) for edge servers, which can select the more and the less priority content to be cached or evicted respectively.

3.2 Fuzzy logic

Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth which denotes the extent to which a proposition is true. While classical logic holds that everything can be expressed in binary terms (0 or 1, black or white, yes or no), fuzzy logic replaces Boolean truth values with a degree of truth. The degree of truth is often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision [52].

A Fuzzy Inference System consists of an input phase, a processing phase and an output phase. In the input phase, the inputs are mapped to an appropriate membership function with specific values. The processing stage consists in performing each appropriate rule and in generating a corresponding result. It then combines the results. Finally, the output phase converts the combined result back into a specific output value. The membership function of a fuzzy set represented by divided ranges defines how each value in the input space is mapped to a membership degree. The inference system is based on a set of IF-THEN statements representing logic rules, where the IF part is called the ”antecedent” and the THEN part is called the ”consequent”.

The fuzzy inference system (FIS) that we built, in our work, consist of four inputs:

1. Mobility that refers to the distance between the mobile user and the nearest base station that contains the cache.
2. Frequency that represents the popularity or how often the contents are requested in a specific period of time interval.
3. The size of the cache that is determined thanks to cache occupancy.
4. The cost of retrieval that reflects the cost of the bandwidth required by a MECS to extract the content as requested by the end user.

Appropriate rules are applied during the processing phase that generates the corresponding results; and finally, producing a quantifiable result with the assignment of the corresponding membership degree. The aim of using fuzzy logic is to take into consideration the environment and the mobile user factors that affect caching performance to make the caching policy context-aware.

3.3 Reinforcement learning

Efficient mobile edge caching policy needs not only to be context-aware, but also to be intelligent to grasp both the environment and the user behavior in order to take optimal or appropriate caching decisions over time. As one of the intelligent learning methods, the reinforcement learning (RL) enables agents to deal with decision making problems by learning through interactions with the environment [3] [28]. As described in [3], the basic components of reinforcement learning enrolment (see figure 2) are the following:

- In the reinforcement learning model, an agent learns through its perpetual interaction with the environment. At each time t and in a state space S, the agent observes a state S_t related to its environment, and from an action space A the agent selects an action a_t according to a policy $\pi = \mathrm{P}(a_t|\mathrm{s}_t)$ which is a probability of choosing action a_t from state s_t.
- The agent earns a reward r_t and moves to a new state S_{t+1}, in respect to the model or to the dynamics of the environment, for reward function $R(s,a)$ and state transition probability $\mathrm{P}(S_{t+1} \mid S_t, a_t)$. The reward is defined as a return $R_t = \infty\, k = 0\ \gamma^k\ r_{t+k}$ where $\gamma \in [0, 1]$ is a discount factor.
- The agent aims at finding an optimal policy, π^*, to achieve from all the states the maximum expected return. The state-value function $V\ \pi(\mathrm{s}) = \mathrm{E}[R_t \mid s_t = s]$ and the action-value function $Q\ \pi\ (s, a) = \mathrm{E}[R_t \mid s_t = s, a_t = a]$ can measure how π is good. $V\ \pi$ (s) represents the expected return from state s and for policy π while $Q\ \pi(s,a)$ represents the expected return when selecting action a in state s and following π.

Reinforcement learning based caching policies have been studied in many works, since it is capable of learning the environment and making online decision through observations. Thus, it is a potential solution to decision problems under dynamic environment, such as the caching policy design problem in [25]. Specifically, a Q-learning based caching scheme was developed in [22] modeling content popularity as Markov processes. while in [51]the authors applied Q-learning to manage named data networking caching. A more realistic approach have been proposed in [54] where the authors design a content caching strategies in mobile D2D networks using multi-agent reinforcement learning without assuming the knowledge of content popularity distribution.

A policy gradient reinforcement learning based caching scheme was introduced in [23] by considering the Poisson shot noise popularity dynamics. Using both service cost and popularity, a dual-decomposition based Q-learning approach was presented in [24]. Particularly, related studies on edge caching, such as [26] [27], have shown that reinforcement learning is effectively labor to be effective in joint resource management.

The caching policies adopted by the RL are trained with observations, depending basically on a reward function resulting from its actions. This reward covers a wide range of factors that can influence the performance of the caching system. The caching policy can also be configured by using RL. In this paper, an emphasize is given to combine fuzzy logic and RL. In turn, the policy of caching can be made more adaptive with the proposed design of reward function while proposing the modifications. The highlight of the combination is to prepare an optimal caching decision. The optimal caching is indicative towards multi-objective optimization of cache decision influencing several parameters (mobility, cost, size and frequency).

4 Our proposal: a modified Reinforcement Learning (mRL) over the edge caching system

Our aim, in this paper, is to propose a modified reinforcement learning (mRL) by combining RL and Fuzzy logic as an embodiment of the idea of exploiting previously acquired knowledge and capabilities from others in order to speed up learning in RL. For example, the knowledge about the content distribution, the mobile user properties and the network conditions can be used by the caching agent. The combination of knowledge methods and RL has significant advantages like constructing more efficient caching policies. On one hand, transferring prior knowledge can be used to help training RL agents, thus making convergence to the optimal caching decision easier. On the other hand, using policies learned by other relative networks with RL agents will improve the efficiency and the robustness of the current RL algorithm.

In this section, we focus on the scenario of caching contents in edge nodes, as depicted in figure 3. There are C contents, denoted as $c_i = \{1, ...C\} : c \in C$, we note that all mobile users in the system may request R requests. The content popularity is defined as fr, which is the frequency distribution of content requests from all users. S is the content size denoted as: $s_i = \{1, ...S\} : s \in S$, the user mobility is modeled by the distance between the user and the nearest base station and finally the cost of retrieving the requested content. The contents distribution is described by MZipf distribution function. For each request, the RL agent in the edge node can make a decision to cache or not to cache according to the decision of fuzzy rules, and if yes, the agent determines which local content shall be replaced if there is no storage space. We assume that content distribution, user mobility, cost and average arrival time of the requests are assigned to each content during t time period. The most important part is how to define a reward function because it affects directly the performance of the algorithm. In order to design a suitable reward function, the key ideais to assign high reward values to content caching actions in order enhance the cache hit ratios. While the loss is the amount of misses, the gain is the amount of cache hits to be enhanced by the newly cached content.

We illustrate an example of applying mRL to the mobile edge caching, where one MEC server is considered and the storage space of the edge cache is initialized to be enough for caching half of the available contents in the network. The MEC server can serve all the requests directly, according to the arrival time. Initially, the caching policy caches locally the contents according to the priority accomplished by the fuzzy system and to the cache size availability. Otherwise, the cache replaces the content less prior by the highest one. Our aim is to find the optimal caching decision by maximizing the cache hit ratio, that is, the number of contents answered by the edge cache. This problem can be solved based on mRL which requires training an agent for representing the policy, and an appropriate reward function that describes how the agent ought to behave. In other words, reward functions have normative contents that provide what the agents should accomplish.

The detailed features are shown in figures 4, 5 and explained below :
The environment is modelled thanks to a stochastic finite state where the inputs are the actions sent from the agent (i.e., fuzzy rules and user requests) and where the outputs are the observations and rewards sent to the agent:

- State transition function $\mathrm{P}(S_{t+1} \mid S_t, a_t)$
- Observation (output) function $\mathrm{P}(O_t \mid s_t, a_t)$
- Reward function $R_t = \infty \, \mathrm{k}{=}0 \, \gamma^k \, r_{t+k}$.

It is noted that the agent observations depend on his actions, which means that perception is an active process. The agent is a stochastic finite state machine having as inputs: observations and rewards received from the environment, and as outputs : actions sent to the environment.

- State transition function: $S_t = \mathrm{f}(S_{t-1} \; O_t, R_t \; , a_t)$
- Policy/output function: $\pi = \mathrm{P}(a_t | \mathrm{s}_t)$

The agent's goal is to find the optimal policy and the state function so that to maximize the expected sum of discounted rewards.

4.1 Mathematical model

In case of caching decision with fuzzy policy dynamically, the learning rate should be modified. However, the modification of learning rate may alter the conventional method of *stochastic gradient descent* (SGD) [48] considerably. We represent the weight adjustment of mRL as follows:
new_weight = (existing_weight – learning rate $\times gradient$), *i.e*, :

$$\theta = \theta - \alpha \frac{\delta}{\delta_{\theta'}} \zeta \theta' \tag{1}$$

with ζ is the tendency of SGD. if α is too small, the gradient descent can be slow. If α is too large, the gradient descent can overshoot the minimum. It may fail to converge or even to diverge. We define a cost function for a neural network. The goal is to minimize this cost function. Hence, it becomes here an optimization problem using SGD. Mathematically, if the cost function (or loss function) is *L(w,b)*, then the goal is to minimize *L*. For a convex optimization problem like this, we use the derivation of the loss function ΔL. Thus,

$$w = w - \eta \Delta L \tag{2}$$

while η is the learning rate.

To consider this principle, we initiate certain modifications of η in the reinforcement learning, which will be a considerable modification over SGD. Precisely, we will modify the following parameter for mRL with respect to SGD: *momentum :* That will help normal SGD by adding a function of the direction of the previous step to a current step. The function is usually a range of [0.1]. Hence, the proposed model of RL can make use of the following configuration changes of learning:

- The learning rate will select new learning values at the end of each epoch/session combining statistical optimization and rejection strategy.

We provide further the following modification :

1. Input search space *S*.
2. Find a mean function μ and variance function ρ
3. η_m is Modified by optimizing α, a function therefore:

$$\sigma_m = \underset{\eta \in S}{\alpha}(\eta; \mu(\eta), \rho(\eta)) \tag{3}$$

5 Problem formulation and model development

5.1 High level description of mRL

The reinforcement learning based algorithms have been successfully applied to various optimization problems in many domains. However, in order to obtain better solutions for specific optimization problems like our multi awareness caching system. The core of the modified RL algorithm used in this study is to generate a sub-environment based on fuzzy policy as following:

- Fuzzy policy weight : F_t, η_m.
- Input search space *S*, the number of initial search
- Update the value of toward an optimum value α^*

We minimize the expected value of the objective in next dynamic instances which implies:

$$policy[f(\theta_t) = policy[f(\mu(\theta_{t-1}), \alpha_t)] \tag{4}$$

We update the previous learning rate (α_{t-1}) to the final value α_t. Then, we perform the gradient descent:

$$\sigma p = (fo\mu(\theta_t, \alpha_t)/(\eta\,\alpha_t) \tag{5}$$

We also assume an optimum value of the learning rate and that at each step the descent value will change towards all the sessions. However, we need to introduce a continuous function with the context of noise, hence:

$$\alpha_t = \alpha_{t-1} - \beta\Delta_\theta f(\theta_{t-1}) + \Delta_\alpha \mu(\theta_{t-2}, \alpha_{t-1}) \tag{6}$$

The rule can be interpreted as multiplication. This will be an additive adaptation of changing of λ.

5.2 Policy improvement

The caching policy can be defined as a set of dynamic artifacts [49]. This phenomenon becomes obvious as the context of caching particularly should be dynamic depending on the condition of network traffic. Hence, the modification of RL also demands essential formal model to cope up with respect to dynamic policies.

5.2.1 Formal model of policy improvement

We define a finite state space S for caching policies including all the possible actions as the policy of caching is stochastic. Therefore, to measure the random changes of the state space of caching denoted by S_c, we introduce a transaction kernel function. For simplicity, we assume a finite set of caching policies and we have as follows:

$$\{K_{sc} : \tau \to [0,\infty]\} < \infty \tag{7}$$

where K_{sc} is the kernel state space for caching.

In the dynamic context of caching, we may have more than one context. This additional context either can be combined or mixed to derive the improved policy. Let there are two such transitional finite kernels to make decision for caching from S_c to τ. Therefore, to evaluate these two kernels, we need to multiply them. Hence, the final policy (as improved) can be formed:

$$K^Q_{sc1} \otimes K^R_{sc2}(S_Q, A) = \int_\tau K^Q_{sc1}(S, dt).\int_u K^R_{sc2}(S_R, t)du \times 1A(t,u) \tag{8}$$

- If the search space is predefined from a stochastic caching mechanism from τ to u for all $t \in \tau$ $u \in$U.

- If the policy for both kernel K^Q_{sc1} and K^R_{sc2} can be combined by integration,the policy π chosen randomly for caching decision is known as greedy.

Therefore, if there is a caching context with all four parameters, then obviously we will select a policy that gives a maximum reward. Therefore, the estimated policy for caching decision to a particular network content is:

$$P_E = \frac{N_E}{Totalnumberoftrainingparameters} \tag{9}$$

In equation (9), N_E is the number of time this policy has been used and the training parameters are out of our four parameters. Among these parameters, the one that has the highest gain at a particular instance allows us to find the number of iterations that the same policy can trigger; which will assist us in decision making.Hence, as a high level step, we formulate the following algorithms:

Algorithm 1 Policy iteration scheme

$fr \leftarrow 0$;
$P_E \leftarrow 0.5$;
trails $\leftarrow$P$_E \times$ntrails;
for (int i = 0; $i \leq$ntrails; i++)
$trails \leftarrow$P$_E \times$ntrails;
end for
If ($fr \leq$trails)
detect greedy ();
fr ++;
else
select A();

Algorithm 2 mRL Caching policy for edge node

Input: Parameters of the system including dynamic conditions: Mb, Fr, CO and Cr
initialization: initialize the network model: content and requests, fuzzy rule table
Learning Loop
Choose an action a_t from an action space A following the policy based on fuzzy caching system π = P($a_t|$s$_t$)
Measure the delay of downloading and cache hit ratio according to (A1)
Apply greedy policy iteration scheme according to (A2)
Update fuzzy rule table
Generate the new caching policy
EndLoop
Cache content based on the optimal policy

5.2.2 Reward function

Recently, there has been novel technologies that replace prediction with a much more efficient way known as reward functions [50]. Reward functions are used for reinforcement learning models and allow to obtain the final results as a conclusion instead of prediction. For decision making problems, prediction is not the only input as the other fundamental input is judgment.

For our caching system, *mRL* treats learning and content placement/replacement as a whole operation. The caching policy adopted by the mRL agent is trained with observations, based on a reward resulting from its actions that relate to the factors that affect caching performance such as mobility. Usually, this reward should be chosen in a way that covers a wide range of factors that can affect the performance such as offloaded traffic or QoE.

Finding the best reward function to reproduce a set of observations can be implemented by the maximum likelihood estimation, Bayesian, or information theoretic methods. The system reward represents the optimization objective. In our work, the objective is to maximize cache hit ratio. For our caching approach, we have all the variables that can go into known values. The time of retrieving is determined while distributing the contents and the users requests (including the associated parameters of each). In the scenario of mRL, when the system state $s(t)$ and the system action $a(t)$ are given, the priority can be determined. When the content requests of a typical user arrive, the system can acquire the knowledge from the fuzzy rules, whether the cache should preform the operation of placement or replacement and also satisfies the requests. We defined a network model where several MECs are deployed. At the network edge, the cache characterized by limited resources is connected to the cloud via the backhaul link. Let $C = 1,2,\ldots,c,\ldots,C$ where C denotes the content provider which is located in the cloud. For simplicity, each cache can store $X(X \leq C)$ contents. Meanwhile, a time-slotted system is considered. Let $N = 1,2,\ldots,n,\ldots,N$ with N the users served by the MECS during time slot t. The distribution of user requests and contents is determined by the ZIPFs distribution function [46]. Let $d_c(t)$ denote the amount of the instantaneous user requests towards C contents during time t. Let $I_t(t)$ denote the cache indicator for the considered MECS during time slot t. Specifically, I_c (t)=1 (with a corresponding priority) indicates that the cth content is cached during the time slot t and $I_c(t) = 0$ otherwise.

Correspondingly, the caching decisions are made according to the fuzzy caching system where the *cache hit rate* measures the caching performance. Hence, it is used as a reward function in the mRL – caching solution.

In the conventional reinforcement learning, an agent learns how to optimize its behaviors in uncertain environment by executing control polices experiencing the decision of rewards and improvising the policy based on the reward. Without satisfyingly strong reward function or decision, the learning may look very difficult for the present problem of caching the state space (combination of mobility, cost, frequency and cache size) that could be too extended. Therefore the time of retrieving information should be maximum and the learning would be difficult. This type of cases may shape the reward decision of caching as extremely sparse. We can approach three different reward decisions on caching itself as in the following:

1. A reward decision with the sparse value that is multiplied with certain values compared to the goal state. However, this kind of reward decision may also slow down the learning because the agent needs to achieve many actions before getting the decision of good caching or not effective caching

2. Reverse reward in case of collisions where the constraints of caching may suffer from the situation where keeping cost always lower cannot optimize the other values of constraints.

3. Zero reward for any other state (ideally, there may be some situations where no decision can be made satisfying the cost as lower).

The reverse reward can be a primary objective in terms of evaluation of the distance and the time of information to be propagated from an initial base station to a nearest base station. It has been observed that collision may happen not for the initial iteration, but when both the number of iterations are increased and either of the three parameters like Mb, Fr and size of the cache are made dynamic. In most of the cases, the designing of the reward function in terms of decision is amalgamated with the respect to the procedure of designing state space. For example, the cache is a time depending problem. Therefore, the distance covered to reach the nearest base station can make a good reward decision or not effective reward decision. To simplify, we only mentioned a generic reward function customized with the present problem:

- The state value function $v\pi(s)$ gives the long term value of state S when following the policy π
- The action value function $q\pi(s,a)$ is the expected return starting form state s taking an action a as in the following:

$$v\pi(s) = X_a \in A\pi(a|s)q\pi(s,a) \tag{10}$$

The expected return of decision for caching will be dynamic for caching process. This is because the state of constraints except the cost will change from time to time. This relation is composed with state value function $v\pi(s)$ and the action value function $q\pi(s\prime,a)$.

In the given expression below, A belongs to the all number of actions and X denotes the all number f states to achieve the based reward decision of caching.

5.3 Data and result analysis

In order to validate our work, a catalogue of N= 10^3 contents has been generated with a cache size of 600MB, where each content is associated with a random size. The frequency is calculated depending on the catalogue size and the number of requests by a user. The cache occupancy is defined as the proportion of the content size among the cache size.

With U number of users $U = \{u,\ldots u_N\}$: $u \in U$, the mobility is formed as the distance $d = XY$ between the users in the points $X = \{x_1, x_2 \ldots . x_U\}$ and the base station in the point Y.

$$Mb = |Y - X_U| \tag{11}$$

The cost to retrieve each content is the time incurred to retrieve the content to the end user. This time is calculated according to the max bandwidth required to provide peak data rates of at least B = 0.2 Mbit/s, and the distance d= XY between the end user and the base station where the cache is located.

$$Cost = \frac{Mb}{B} \tag{12}$$

A resource from the catalogue content is an object that contains an ID, a size and a payload that refer to the type of the content we have (video, image, audio). For simplicity, it is set to a string value. Studies have shown that *Zipfs* law is an appropriate distribution model for the distribution of requests and contents over Internet such as YouTube videos or peer-to-peer file sharing systems. According to *Zipfs* law, most user requests target a small fraction of popular web contents. *Zipfs* distributions related to a catalogue of N contents with request probabilities $Z(r)$ have been defined for the objects popularity ranks as in the following:

$$Z(r) = \alpha r^{\beta} with : r^{\beta} < 0; \beta = Z(1) = 1/\sum_{r=1}^{N} r^{\beta} > 0 \tag{13}$$

where α is a normalization constant and β an adaptive shape parameter. $R = 10^3$ corresponds to the users requests with α set as : $\alpha = 0.75$.

Based on this distribution law, several characteristics curves can be generated (see appendix for the snap of the data). The characteristics of the curves are defined as following

- Choosing an optimization function to balance the four optimal parameters like frequency, mobility, size and cost. However, the cost as well as the mobility always have to be kept as minimum as possible.
- Deviation of cache size and time of caching is expected.
- The optimization of objective function to balance mobility and cost with the respect to the others generates another section of optimization
- It is also expected that there will be reference mean and an actual mean between the max and the min values of bandwidth. Hence, setting histogram plots could be worthy to demonstrate these variations.

The next subsection describes the analogy behind the data and it demonstrates also the four given components as a holistic performance of the model followed by an optimization characteristic curve.

5.4 Results Discussion

We should mention that unbalanced data have been used for two reasons. First, the scenario is a multi-objective [47]. Second, using unbalanced data requires methods to solve the problems of optimization under nonlinear constraints of inequalities. Hence, we used Karush-Kuhn-Tucker conditions (KKT) which is an optimization problem with interval-valued objective function

[44] [45]. For general characteristics, the deviation of cache size may occur with respect to time of caching (see figure 6). In this figure, there are two segments of curves. The one in red color demonstrates that the cache size is 0.01 unit and the time of caching is only variable (in ms), then the slop of the curve becomes linear and deviates more toward a sustain value. After 3 ms, there is no significant change in deviation of cache size. However, in the same figure, the upper segment shows cache size between [0.02-0.03] units. If it differs, then the slop of the curve can only be flat after 25 to 30 ms time. This describes that the deviation of cache size while keeping the time of caching fixed can be a significant characteristic to the performance of this model. In the histogram plot of figure 7, frequency is a range of 10^3 samples. However, these large samples of ranges of frequency may not serve the RL model as part of the given data shown in appendix. It also shows that the axis x of the plot of histogram can be sustained and becomes non significant after 20000 values of Fr. Therefore, there is a requirement to bifurcate the maximum and the minimum range of frequency by using the actual mean value as a reference as shown in the plot. This will help to generate the data proposed for this model accordingly.

After formulating the data generation and investigating the tendency of cache size with time, the given performance has been focused on a multi objective optimization problem. Here, the four parameters, used to analyze the trusted caching decision, can vary and significantly impure the results. For example in figure 8, it is shown x axes containing an iteration count and y axes containing an optimization function. In this case, the optimization function can vary with

cost and mobility, although two variables are kept as minimum with respect to the remaining two variables. Finally, this will produce at least two optimization functions with respect to cost and mobility. Therefore, the data part should be unbalanced to yield an optimal solution of caching. It is shown in the figure that the training data in this model has been occurred from fuzzy model and therefore there may be substantial possibilities to present data in a mixed cluster of any of these four variables. These features provide that there is a necessity of final clustering on the unbalanced data for optimal caching decision. The figure has demonstrated three dots where clearly the middle part shows dense clusters of any of the four variables line in this region. However, the left and right sides of those dots are clearly isolated from these clusters, which means that the middle dense cluster can be suitable for a good caching decision whereas the other may not. These all dots represent the combination of any of the four parameters without any balance ratio. This necessitates the requirement *KKT* optimization in the analysis part of these data. We followed a simple linear regression model according to Figure 9. In this figure, it is a three dimensional plot to show the relation between cost and frequency and it is shown in the below part of the three dimension plot that regression becomes dense. However, the upper part becomes scattered and we observe that the generated data have not been balanced enough to produce a good outlier decision for caching. Outlier means that each time the data and the iterations of caching session should be proportional. Data is the combination of four parameters as it is unbalanced. Therefore, the relationship between cost and frequency may not provide any significant decision for caching. Additionally, it can be noted that for bandwidth or frequency, we have referred the histogram plot for reference mean and actual mean of the maximum and minimum frequency. Therefore, we require more optimal clustering for Figure 10. Here, we observe that there are three axes in the given plot where Fr, CR, and CO of the content are plotted based on the sample data generated (see appendix). In the given figure, there are three sections based on the distribution of the optimization function:

- The dense blue dots occupy the Fr and the CR region of the plot. This entropy will distribute certain scattered blue dot from CR toward the mapping region of the cost. As it is mentioned, the cost should be kept as minimum. Therefore, very few blue dots are available in the cost region satisfying the variable Fr and CR according to the generated data.
- This plot is a broader outlier analysis of clustering. It describes the plot regions into different visible sections. However, due to the absence of real effective cost data, it is not possible to find more blue dots which may represent appropriate cost of caching with respect to the other parameters.
- The modified RL thus can distribute a map value for state - action and reward clustered for appropriate decision of caching.

In both figures:

- 3D distribution of input data has been shown using scatter diagram. Additionally, these following values are analyzed :
  - Means, Median, Std/Variance all attributes, data distribution and histograms.
  - Co-variance of pair of attributes.

Pairwise linear regressions.

- $Fr = a + b \times Cr$.
- $Fr = a + b \times Co$.
- $Fr = a + b \times Mb$.
- $Cr = a + b \times Co$.
- $Cr = a + b \times Mb$ and $Co = a + b \times Mb$
- Multiple linear regressions ($Fr = a + b \times Cr + c \times Co, Fr = a + b \times Cr + c \times MB, Co = a + b \times Fr + c \times Mb, Fr = a + b \times Cr + c \times Co + d \times Mb$)

- Multiple quadratic regressions $Fr = c_0 + c_1 \times Cr + c_2 \times Co + c_3 \times Mb + c_4 \times Cr \times Cr + c_5 \times Co \times Co + c_6 \times Mb \times Mb + c_7 \times Cr \times Co_c 8 \times Cr \times Mb + c_9 \times Co \times Mb$

To identify possible clustering, results are shown using graphical representations. For all regressions, we have used Least Square optimization techniques for the learning of model constants, i.e., regression coefficients.

Scatter diagram is showing the distribution of the data in all directions. There is no such biased or prominent grouping of data. From the graphical and tabular views, it can be easily found that some models are working with certain errors. This is also to emphasize that apparently it seems to formulate a perfect optimization function based on 4 parameters. However, in real time, it may not be suitable too.

6 Conclusion

In this paper, we have proposed and demonstrated a realistic decision scheme for caching scenario by using reinforcement learning as a machine learning component. Hence, to incorporate RL in the mobile edge caching system, we proposed certain elementary modifications. This is because following the principles of RL, the state, policies and actions or rewards are continuous variables inter-playing with each other. To consolidate this objective, the present content of the paper has been initiated with fuzzy logic module [20] followed by the inclusion of RL. This is because in normal fuzzy system, the values of membership becoming static in certain cases, may not follow the continuous function. Therefore, the formulation of a hybrid model with fuzzy and RL is required. However, the modification were expected to customize mobile-edge parameters (frequency, mobility, size and cost) while the respective policies should be kept dynamic. These dynamic properties of the proposal can support the different caching scenarios at different intervals depending on the demand and the hit ratio in real time. Certain interesting observations are made from the given proposal. It is demonstrated in the paper that to keep a minimum cost for the whole system and to vary different other parameters except mobility, a well balanced machine learning based optimization function is formulated. The data snap and the analogy have also been described in appendix of the paper for further reference. Considering the demand of the huge bandwidth and big data, the dynamic and intelligent component could foster machine learning and mobile edge caching systems more contemporary.

Finally, we have presented a decision scheme for caching scenario using a modified reinforcement learning model. Our future work will investigate the use of real datasets generated from real information centric networking scenarios.

References

[1] Wang X, Han Y, Wang C, Zhao Q, Chen X, and Chen M. In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning. ArXiv preprint. vol. abs/1809.07857. Sept. 2018.

[2] Luo Z, LiWang M, Lin Z, Huang L, Du X, Guizani M. Energy-efficient caching for mobile edge computing in 5g networks. Appl. Sci., vol. 7, no. 6, pp. 557, 2017.

[3] Hao Zhu ; Yang Cao ; Wei Wang ; Tao Jiang ; Shi Jin. Deep Reinforcement Learning for Mobile Edge Caching: Review, New Features, and Open Issues. IEEE Network, Volume: 32 , Issue: 6 , November/December 2018, pp.50-57

[4] Botvinick M, Niv Y, and Barto A. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. Cognition, vol. 113, no. 3, pp. 262–280, Dec. 2009.

[5] Niv Y and Schoenbaum G. Dialogues on prediction errors. Trends Cogn. Sci. (Regul. Ed.), vol. 12, no. 7, pp. 265–272, Jul. 2008.

[6] N. D. Daw. Chapter 16 - Advanced Reinforcement Learning. in Neuroeconomics (Second Edition), P. W. Glimcher and E. Fehr, Eds. San Diego: Academic Press, 2014, pp. 299–320.

[7] J. Suomala, V. Suomala. Modified Reinforcement Learning Infrastructure. 2nd International Conference on Applied Social Science Research (ICASSR 2014). pp. 95-97.

[8] E. Bastug, M. Bennis, and M. Debbah. Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks. IEEE Commun. Mag., vol. 52, no. 8, Aug. 2014, pp. 82-89

[9] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, S. Chouvardas. Placing dynamic content in caches with small population. Computer Communications IEEE INFOCOM, pp. 1-9, 2016.

[10] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, Valerie Young. Mobile Edge Computing: A Key Technology Towards 5G. ETSI Whitepaper, September 2015, ISBN 979-10-92620-08-5.

[11] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, M. Guizani. Energy-Efficient Caching for Mobile Edge Computing in 5G Networks. Applied sciences, vol. 7, no. 6, pp. 557, 2017.

[12] M. Zhang, H. Luo and H. Zhang. A Survey of Caching Mechanisms in Information-Centric Networking. in IEEE Communications Surveys Tutorials, vol. 17, no. 3, pp. 1473-1499, thirdquarter 2015.

[13] S. Arif, S. Hassan and I. Abdullahi. Cache Replacement Positions in Information-Centric Network. The 4th International Conference on Internet Applications, Protocols and Services (NETAPPS2015), pp.54-58, Malaysia, 2014.

[14] . Psaras, W. K. Chai, G. Pavlou. Probabilistic in-network caching for information-centric networks. ACM Workshop on Information-Centric Networking (ICN) ,pp. 55-60, 2012.

[15] J. M. Wang, J. Zhang, B. Bensaou. Intra-as cooperative caching for content-centric networks. Proc. 3rd ACM SIGCOMM Workshop ICN, pp. 61-66, 2013.

[16] A. S. M. A. Ibrahim Abdullahi, Ibrahim Badamasi. Cache skip approach for Information-Centric Network. ARPN Journal of Engineering and Applied Sciences, vol. 11, no. 5, pp. 3413-3418, 2016.

[17] Stefan Podlipnig and Laszlo Bszrmenyi. A survey of Web cache replacement strategies. ACM Computing Surveys(CSUR), Volume 35, Issue 4, pp. 374-398, December 2003.

[18] M. Osman, Areej Osman, Niemah. (2018). A Comparison of Cache Replacement Algorithms for Video Services. International Journal of Computer Science and Information Technology. pp 95-111, 2018.10208.

[19] S. Mehamel, K. Slimani, S. Bouzefrane, M. Daoui. Energy-efficient hardware caching decision using Fuzzy Logic in Mobile Edge Computing. The IEEE 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloud'2018), August 2018, pp.237-242, Barcelona, Spain,

[20] S. Mehamel, S. Bouzefrane, K. Slimani, M. Daoui. New caching system under uncertainty for Mobile Edge Computing. The Fourth IEEE International Conference on Fog and Mobile Edge Computing, June 2019, pp.to appear., Rome, Italy,

[21] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas. Placing dynamic content in caches with small population. in Intl. Conf. Comput. Commun., San Francisco, USA, April 10-15, 2016, pp. 1–9.

[22] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis. Optimal and scalable caching for 5G using reinforcement learning of space-time popularities. IEEE J. Sel. Topics Signal Process., vol. 12, no. 1, pp. 180–190, Feb. 2018.

[23] S. O. Somuyiwa, A. Gyorgy, and D. Gunduz. A reinforcement-learning ¨ approach to proactive caching in wireless networks. IEEE J. Sel. Areas Commun., vol. 36, no. 6, pp. 1331–1344, June 2018

[24] A. Sadeghi, F. Sheikholeslami, A. G. Marques, and G. B. Giannakis. Reinforcement learning for adaptive caching with dynamic storage pricing. pp, 2018

[25] Chen, Jiayin Xu, Wenchao Cheng, Nan Wu, Huaqing Zhang, Shan Sherman Shen, Xuemin. Reinforcement Learning Policy for Adaptive Edge Caching in Heterogeneous Vehicular Network. pp 1-6.(2018)

[26] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis. Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities. in IEEE J. Sel. Top. Signal Process., vol. 12, no. 1, pp. 180-190, Feb. 2018

[27] 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.

[28]] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. Cambridge, MA, USA: MITPress, 2016.

[29] F. Zhang et al. Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture. Proc. IEEE 16th Int. Symp. World Wireless Mobile Multimedia Netw. pp. 1-9, Jun. 2015.

[30] A. Mahmood, C. Casetti, C. F. Chiasserini, P. Giaccone and J. Harri. Mobility-aware edge caching for connected cars. 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS), Cortina d'Ampezzo, 2016, pp. 1-8.

[31] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao and X. Shen. Cooperative Edge Caching in User-Centric Clustered Mobile Networks. IEEE Transactions on Mobile Computing, vol. 17, no. 8, pp. 1791-1805, 1 Aug. 2018.

[32] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. in Proc. of the second edition of the ICN workshop on Information-centric networking, pp.55-60, 2012.

[33] D. Rossi and G. Rossini. Caching performance of content centric networks under multi-path routing (and more). Tech. Rep., Telecom ParisTech, 2011.

[34] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack. WAVE: Popularity-based and collaborative in-network caching for contentoriented networks. Proc of IEEE INFOCOM WKSHPS 2012, pp.316-321, March 2012.

[35] Stefano Salsano, Andrea Detti, Matteo Cancellieri, Matteo Pomposini, Nicola Blefari-Melazzi. Transport-Layer Issues in Information Centric Networks. ACM 978-1-4503-1479-4/12/08;Agust 17 2012,

[36] Nikolaos Laoutaris , Hao Che , Ioannis Stavrakakis. The LCD interconnection of LRU caches and its analysis, Performance Evaluation. v.63 n.7, p.609-634, July 2006.

[37] Xiaohu Chen,Qilin Fan ; Hao Yin. Caching in Information-Centric Networking: From a content delivery path perspective. Innovations in Information Technology (IIT), 2013 9th International Conference on on Innovations in Information Technology;IEEE;10.1109/Innovations.2013.6544392 page 48-53.

[38] T.M. Wong, J. Wilkes. My cache or yours? Making storage more exclusive. in: Usenix Association Proceedings of the General Track, 2002, pp. 161–175.

[39] D. Mardham, S. Madria, J. Milligan and M. Linderman. Opportunistic distributed caching for mission-oriented delay-tolerant networks. 2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS), Isola, 2018, pp. 17-24.

[40] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, S. Nanda. Congestion-aware caching and search in information-centric networks. Proc. 1st ACM Int. Conf. Inf. Centric Netw. (ICN), pp. 37-46, Sep. 2014.

[41] J. M. Wang, B. Bensaou. Progressive caching in CCN. Proc. 31st IEEE Glob. Commun. Conf. (GLOBECOM), pp. 2727-2732, Dec. 2012.

[42] L.Kwong Yuen, T.Zahir and B.Peter. Supporting user mobility through cache relocationMobile Information Systems. vol. 1, no. 4, pp. 275-307, 2005

[43] W.James Z, Du.Zhidian and S. Pradip K. Cooperative proxy caching for wireless base stations. Mobile Information Systems, vol. 3, no. 1, pp. 1-18, 2007

[44] Hsien-Chung Wu. The Karush–Kuhn–Tucker optimality conditions in an optimization problem with interval-valued objective function. European Journal of Operational Research, Volume 176, Issue 1, 2007,Pages 46-59,

[45] Johannes Jahn. Karush–Kuhn–Tucker Conditions in Set Optimization. Journal of Optimization Theory and Applications, 2017, Volume 172, Number 3, Page 707

[46] Xavier Gabaix. Zipf's Law for Cities: An Explanation. The Quarterly Journal of Economics, Volume 114, Issue 3, August 1999, Pages 739–767

[47] Marler, R.T. and Arora , J.S. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization, 2004, pp 371.395.

[48] Ch.Netrapalli, R.Ge, M. Sham, Kakade and M.I. Jordan. Stochastic Gradient Descent Escapes Saddle Points Efficiently. CoRR journal, vol abs/1905.03776, 2019. pp 1.17.

[49] P.Mannion, S.Devlin, K.Mason, J.Duggan, E.Howley. Policy invariance under reward transformations for multi-objective reinforcement learning. Volume 263, November 2017, PP 60-73.

[50] L. He, Y. Chu and C. Shen. A Design of Reward Function in Multi-Target Trajectory Recovery with Deep Reinforcement Learning. 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 2019, pp. 286-293.

[51] Amar Abane, Mehammed Daoui, Samia Bouzefrane, Paul Muhlethaler. A lightweight forwarding strategy for Named Data Networking in low-end IoT. Journal of Network and Computer Applications, Elsevier, 2019, 148, pp.102445.

[52] Diab Hassan, KashaniAli, Nasri Ahmad. Cache replacement engine: A fuzzy logic approach. Proceedings of the 2009 International Conference on the Current Trends in Information Technology, CTIT 2009.

[53] Sunitha Safavat, Naveen Naik Sapavath, Danda B. Rawat, Recent advances in mobile edge computing and content caching, Digital Communications and Networks, 2019,pp.1-6.

[54] W. Jiang, G. Feng, S. Qin, T. S. P. Yum and G. Cao, Multi-Agent Reinforcement Learning for Efficient Content Caching in Mobile D2D Networks, in IEEE Transactions on Wireless Communications, vol. 18, no. 3, pp. 1610-1622, March 2019.

Table 1: Summary of some existing classical placement mechanisms

Placement strategy	Type	Description
Leave Copy Everywhere (LCE) [35]	Homogeneous, non-cooperative, on-path	caching the content at each node that it traverse along the downloading path what causes caching redundancy.
Leave Copy Everywhere (LCD) [36]	Heterogeneous, cooperative on-path	caching the content one hop closer to the user or to the next hop down, it caching only popular content and prevent the unpopular
Move Copy Down(MCD) [36]	Heterogeneous,cooperative, on-path	caching all popular content, the policy move the copy of the requested content with a hit to its underlying cache in the path and deletes it, it may suffer from caching pollution on the local replica in case of multiple requests
Probabilistic Cache (ProbCache) [37]	Heterogeneous,cooperative, on-path	Contents should be cached closer to their destination with higher probability in order to leave caching space
Intra-AS Cooperative Caching [15]	Homogeneous, cooperative,off-path	allows different caches to serve each other's request leading to solve the limited storage space problem and eliminate redundancy.
Least Unified Value(LUV) [15]	Homogeneous, cooperative, on-path,	Each cached content is assigned a Least Unified Value(LUV) with cache distance from the content source and cache the content later according to this value
WAVE [38]	Homogeneous, non cooperative, on-path	The contents are cached based on their popularity and counts the access frequency of requests.
Opportunistic Caching (OC) [39]	Homogeneous, non cooperative, off-path	Probability based caching policy that use the distance factor beside the popularity factors to cache a content.
Congestion-Aware Caching (CAC) [40]	Homogeneous, non cooperative, on-path	it aims to decrease the download time by exploiting the available cache capacity, it is based on two factors the download time and the content popularity
Progressive caching policy (PCP) [41]	Heterogeneous, cooperative, on-path,	it is a combination of some existing caching policies that are based on the position of a cache in the network : intermediate or edge , PCP avoid caching unpopular contents and shares the characteristics of both LCD and probabilistic caching

Table 2: List of mathematical symbols

Symbol	description
S_t	State
A_t	Action
π	Policy
R_t	Reward function
V	state-value function
Q	action-value function
θ	weight
η	learning rate
σ	Gradient descent
μ	Mean value
ρ	Variance function
F_t	Fuzzy policy
K	Kernal space

Table 3: Notation

Notation	Description
S_i	cache size
Id_i	Id or Index of content
N	Number of contents
R	Number of Requests
C	content provider
U	Number of users
Fr	Frequency of demand of each content in time period
Cr	The time incurred to retrieve the content
Co	Occupancy of content in the cache
Mb	The users proximity from the base station

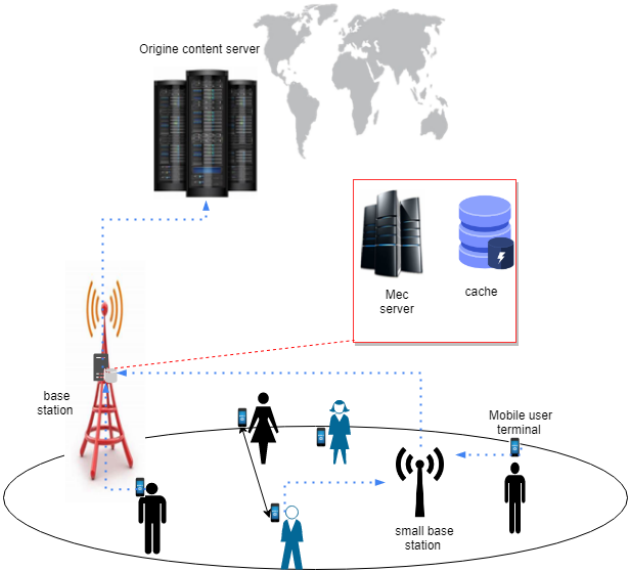

Figure 1: Edge caching architecture

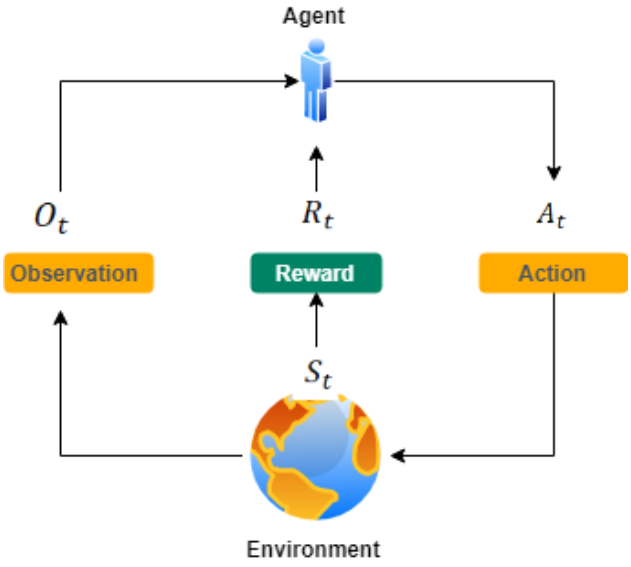

Figure 2: Reinforcement learning model

A
Appendix A

Part of the data :

FR(% request i	CR (seconds)	CO(% of Migal	MB (client IP modeled by distance)
0,7	1	0,01666667	1
1	0,8	0,01666667	0,5
1	0,85	0,01666667	0,1
0,8	1	0,91666667	1
1	1	0,05	0,7
0,6	0,5	0,56666667	0,8
0,55	0,9	0,01666667	1
1	1	0,35	1
0	0,4	0,35	0,8
0,5	1	0,13333333	0,7
1	1	0,03333333	0,45
0,4	0,5	0,05	1
1	0,75	0,05	0,6
0,65	0,6	0,01666667	0,3
0,6	1	0,05	0,3
1	0,45	0,05	0,7
0,4	0	0,13333333	0
1	0,4	0,21666667	1
0,8	0,5	0,01666667	0,2
0,55	1	0,01666667	0,1
0,3	0,7	0,03333333	0,4
0	1	0,05	0,3
1	1	0,03333333	0,1
0,6	1	0,01666667	0,2
0,5	0,9	0,01666667	0,1
1	0,6	0,13333333	1
1	0,9	0,05	0,1
1	1	0,56666667	0,6
0,9	0,1	0,01666667	0,4

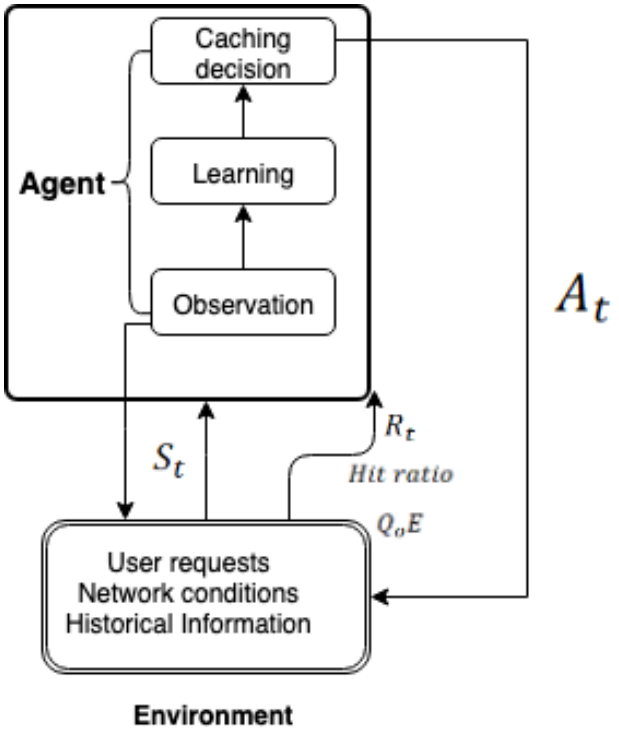

Figure 3: scenario of caching using reinfocement learning

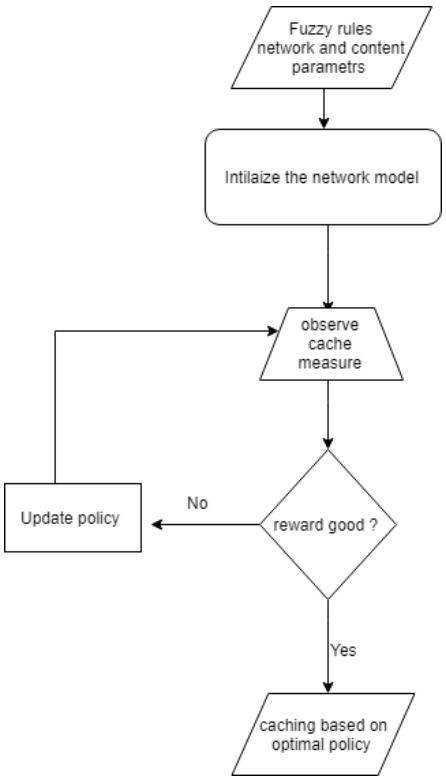

Figure 4: block diagrame of caching over mRL

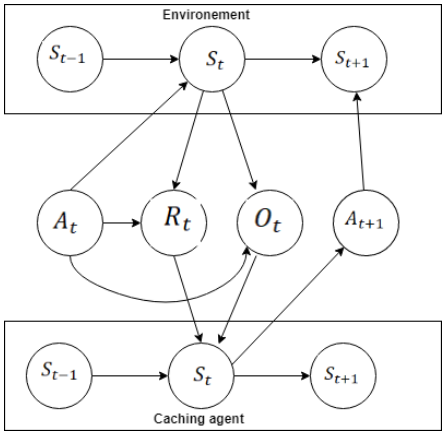

Figure 5: cahing Agnet process

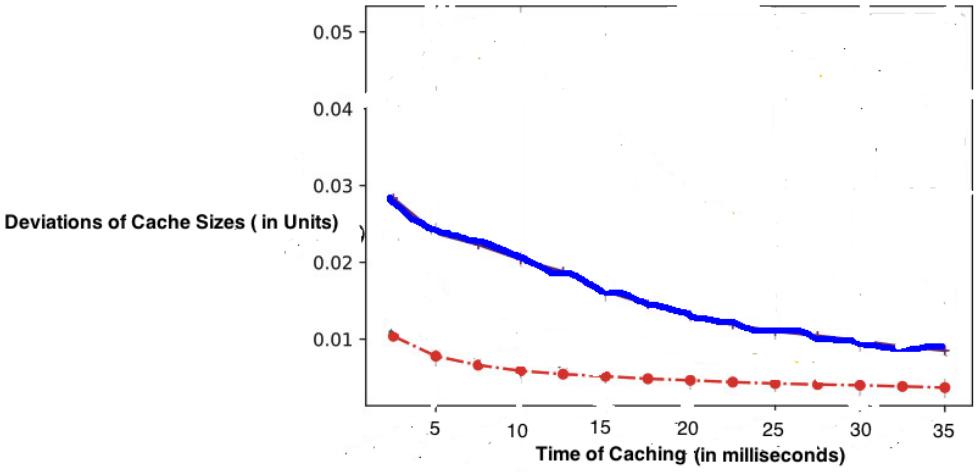

Figure 6: cache size over time

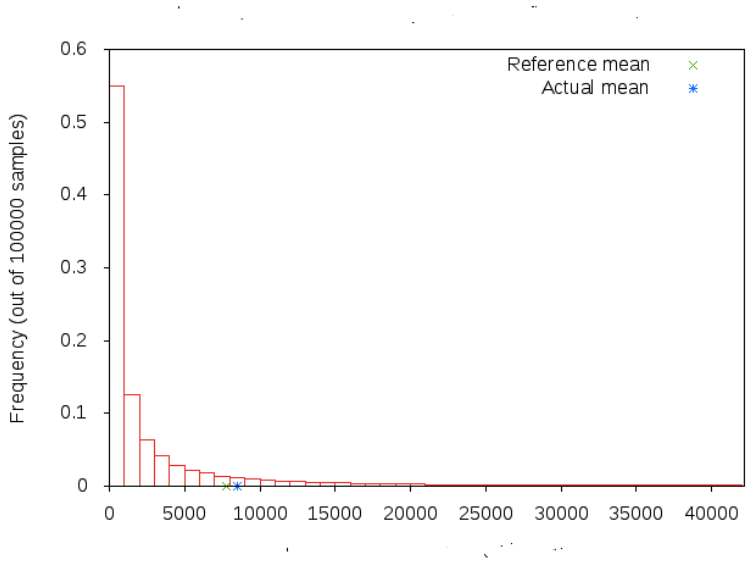

Figure 7: frequency histogram

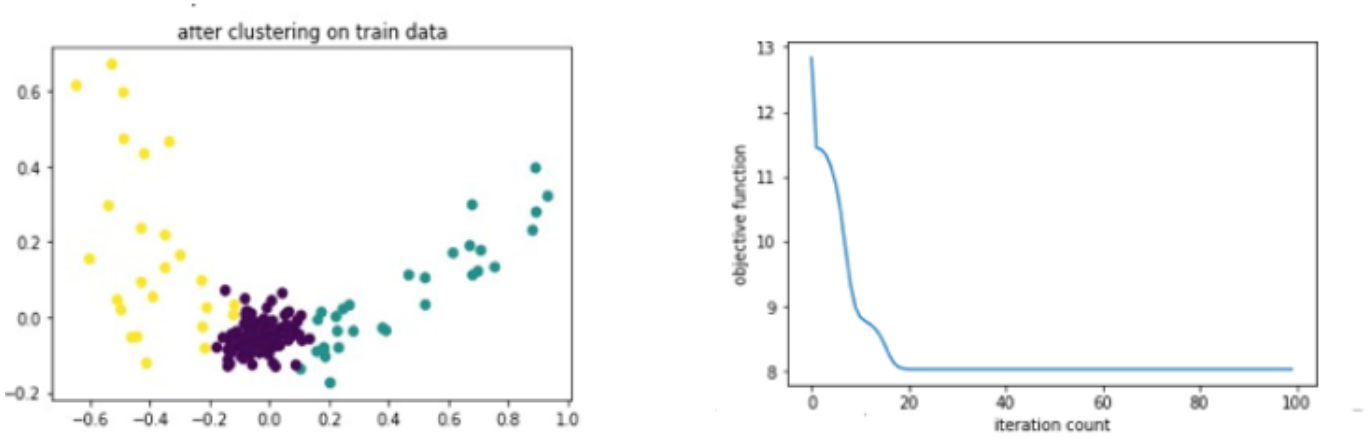

Figure 8: caching decision evaluation and the optimization of the objective function

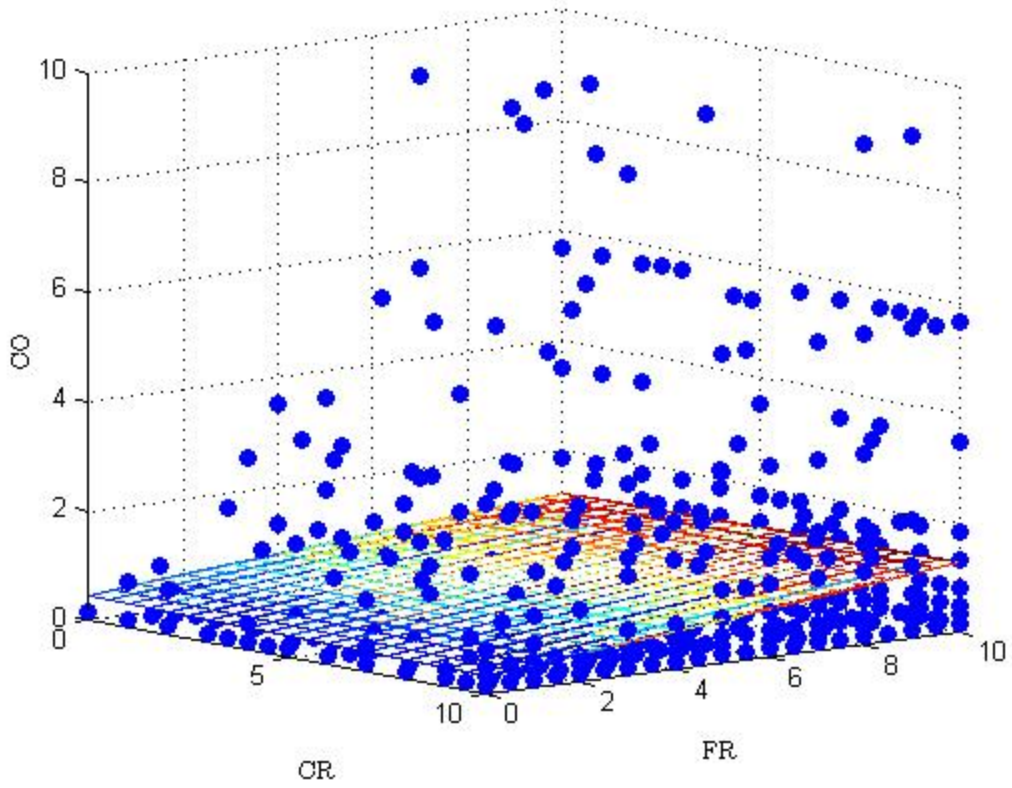

Figure 9: relation between cost, frequency and size

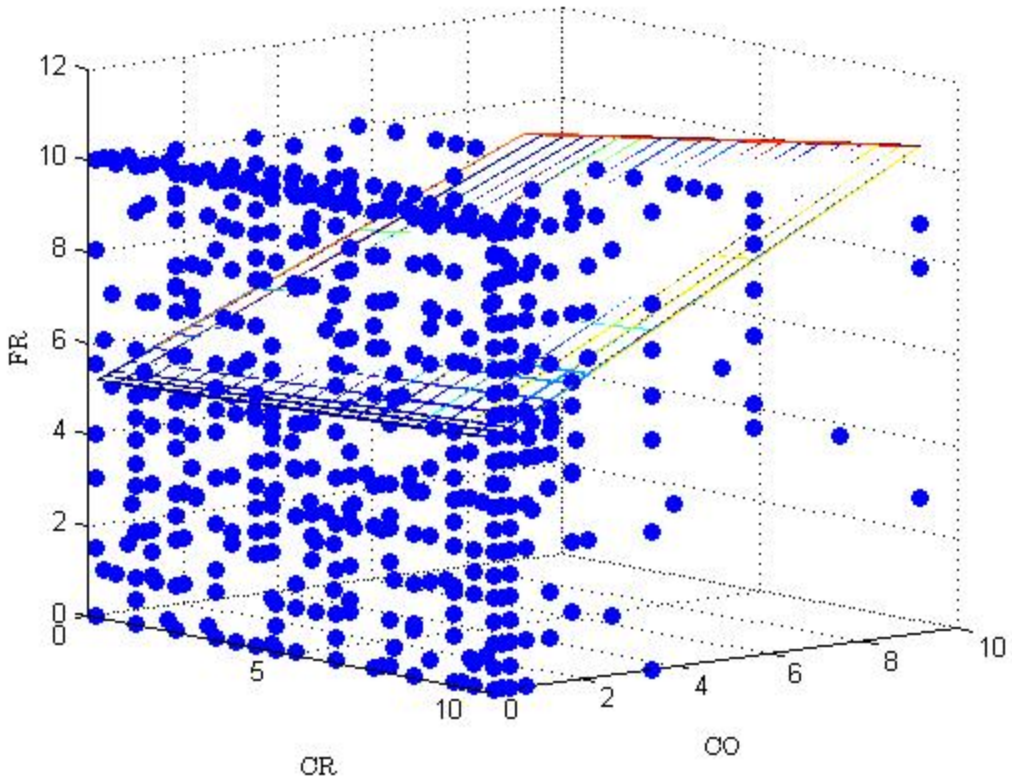

Figure 10: optimal clustering for cost and frequency and size