iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** IV **Month of publication:** April 2022

DOI: https://doi.org/10.22214/ijraset.2022.41256

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue IV Apr 2022- Available at www.ijraset.com

Optimizing Cache Hit Ratio in Edge Computing Using Machine Learning Techniques

Muddasar Khan[1], Prof Wei Wei[2], Falak Naz[3]
[1, 2]*College of Information Science and Technology, Henan University of Technology, Zhengzhou, 450001, China*

Abstract: ***Over the past few years, computing points in cloud computing systems have begun to extend to terminal nodes in network infrastructure as more powerful and smarter devices become available. This extent to computing power has triggered a variety of terminology, including fog computers, Nano data centers, and cloudlets. Edge caching has aroused increased interest in research as mobile traffic grows rapidly and more traffic is devoted to serving content (as opposed to communication between end nodes). By placing popular content in the cache for edge servers that are closer to the end of the user, a wireless network can simultaneously achieve a smaller content delivery delay for the users and an improved spectrum and energy efficiency.***
Keywords: edge computing, fog computers, Nano data centers, machine learning algorithm

I. INTRODUCTION

Mobile Edge Computing the MEC was launched by the European Telecommunications Standards Institute to enable cloud computing services close to mobile subscribers [3]. With the development of MEC servers in macro or micro base stations, MEC can enhance the user experience by editing the user application at the edge of the network with reduced latency and location awareness.

The fog computing paradigm was first invented by Cisco in 2012. In a sense, fog calculations

Similar to the MEC concept. However, it is also a new network computing architecture which provides computing power on the edge of the network. Fog computing was originally proposed for IoT contexts that require location awareness and timely response, in addition to wireless access and mobility support. In addition, fog computing uses an n-tier architecture to provide more flexible services.

The concept of cloudlet was first proposed in 2009 by a research team at Carnegie Mellon University. The term "cloudlet" refers to a nearby micro data center percentage of mobile users such as coffee shops and classrooms. The main motivation behind Cloudlet is to improve the interactive performance of mobile applications, especially those with stringent end-to-end latency and jitter requirements. For such applications, the response delay should be on the order of milliseconds, which may not occur on the Internet.

The concept of cloudlet was first proposed in 2009 by a research team at Carnegie Mellon University. The term "cloudlet" refers to a nearby micro data center percentage of mobile users such as coffee shops and classrooms. The main motivation behind Cloudlet is to improve the interactive performance of mobile applications, especially those with stringent end-to-end latency and jitter requirements. For such applications, the response delay should be on the order of milliseconds, which may not occur on the Internet

We observe that the hit ratios are very low for the proposed caching schemes in edge computing. Further, the proposed works compare their work with basic caching techniques rather than with similar ML based caching techniques. We will apply unsupervised learning with pre-fetching to solve the problem of low cache hit ratio in edge computing.

II. LITERATURE REVIEW

Machine learning is a method that often uses measured data for modeling the actions that a system can perform. In wireless networks it can be used to design caching policies, because it can predict the popularity of the content or its ranking by using the data available on social networks.[2] Data therefore plays a crucial role and makes machine learning a data-based method. In the literature [4], proposed an Aged-Based Threshold (ABT) approach which caches only those contents which have higher request rate than a threshold value. The proposed approach gives optimal results with increased number of contents and gives poor results when number of content is small. Then an architecture is studied for estimation of popularity at; i) Global cache ii) Local cache. Global cache learns faster by aggregating local caches while local caches give better performance. In this study cache hit rate in not calculated. The study in [5] proposed a Deep Reinforcement Learning (DRL) based framework with Wolpertinger architecture for the problem of content caching. In this work a DRL agent is designed for the policy control problem for content caching. In policy control the authors decide that which contents to be stored in the caches. Caching decisions are made at the edge node in this case a base station.

Most frequently requested content are stored in local storage by making caching decisions using DRL agent based on user's request. When a user request a content, first it will be checked in the local cache of the base station. If the requested content exists it will be served directly with minimum delay. If the content is not stored locally it will be requested by the base station from the original server and using caching policy it will update the local cache. Cache hit rate is calculated using different caching capacities. The proposed framework has a high hit rate for all caching capacities. Further the proposed framework is compared with other caching algorithms such as LRU, LFU, and FIFO in terms of cache hit rate. In this study the issue is that; a single base station is considered. All the algorithms in this work have a close cache hit rate around 0.8. In [6] a proactive caching mechanism known as learning-based cooperative caching (LECC) is proposed which uses Transfer Learning approach (TL) for content popularity estimation. Taking into account the dynamic update of the content and the demands of the users, the available historical access information of the specific content element may be very small for its popularity prediction. It is very difficult to estimate content popularity due to data sparsity and cold start issues. To address this issue, the authors use a TL-based approach to predict the popularity of content. Transfer learning uses the knowledge of some related tasks in the target task when it does not have sufficient quality training data. Better learning performance is achieved when there is a great correlation between the target and source domain. Using communications amongst the base stations cached content can be shared between the MEC servers. For the communication between MEC servers X2 interface is used. Evaluation of performance shows that LECC can apparently improve the content cache hit rate and the delay in delivery of content and transmission costs compared to known existing caching strategies.

A. Proposed Framework

We present our proposed mechanism for caching using edge computing and show how our mechanism works. After that, we present our proposed algorithm and show all the included steps. Furthermore, we briefly explain how our proposed algorithm for detecting similar patterns in data. Finally, we present our results. We simulated different users' requests using JMeters and compared our results with existing clustering techniques that are used in content delivery networks and popularity-based edge caching technique. We observed that our proposed mechanism shows better results.

B. Community Detection

For detecting communities in social networks, we have gathered data from twitter through real time streaming API and used the existing dataset from Kaggle. The size of our dataset is one million. After that, we had performed existing community-based clustering techniques on our one million twitter data. By analyzing various pros and cons of existing clustering techniques we have proposed a hybrid clustering algorithm, our proposed algorithm is a modification of Density based Spatial clustering with a flavor of minibatch k-mean that scales well in terms of the size of the dataset, and for detecting large numbers of clusters with minimum processing cost.

C. Data Collection and Preprocessing

We have collected our dataset from the following source:

1) Crawled Tweet Using twitter API
2) Using existing datasets from kaggle

D. Data Pre-processing

In the pre-processing stage, the data is cleared from http links, special characters, blank spaces, stop words, intentions, empty tweets and punctuation marks, etc. are removed from the data. Next, we split the data set into location bins, that is, we split each location tweets. The location bin determines the location of the tweet. In order to represent the data, the data is displayed in the form of graphs.

E. Cleaning Data

Python Libraries such as (Pandas and Numpy) are used.

F. Data Visualization

For data visualization, python libraries such as (Matplotlib) are used.

G. Clustering Techniques

To evaluate the problem, we group the data to find similar data points within the data, which are ignored by previous work except for the K-mediods algorithm. The size of feature is 100 in our work. Based on the processing time, the following grouping algorithms are tested.

- K-Means
- DBSCAN
- HDBSCAN

1) K-Means Clustering

Algorithm

1. **Choose the number of clusters (K) and obtain the data points (X)**
2. **Place the centroids c_1, c_2, ……. c_ k randomly**
3. **Repeat steps 4 and 5 until convergence or until the end of a fixed number of Iterations**
4. **for each data point x _ i : find the nearest centroid(c_1, c_2 .. c _k) assign the point to that cluster**
5. **for each cluster j = 1……k**
   new centroid = mean of all points assigned to that cluster.
6. **End**

K-Means is an unsupervised clustering algorithm that randomly selects *k* starting points as the center of the cluster. In a data set, K-Means finds a predefined number (k) of groups. Using Euclidean distance, each point in the data set is mapped to the nearest center of the group. The center of the team is recalculated again and again based on the average of the points available in the team. Repeat the process until no further change in the group center value is observed. K-Means cannot deal with noise and excessive prices.

2) DBSCAN: (Density-based Spatial Clustering of Application with Noise):

Unlike other clustering algorithms, DBSCAN detects so many clusters in the data. DBSCAN also handles noise and outliers in the data. It works on two parameters.

a) Eps, or "The maximum distance between two monsters to be considered in the same neighborhood" and

b) min_samples, the "number of samples (or total weight) in a neighborhood for a point to be considered a core point.

The epsilon value for Dbscan is (eps_0.9) and the number of minimum points is (min_samples=20) to create a dense area. The goal here is to find out the performance of DBSCAN in finding clusters.

Algorithm

1. **Input: A dataset containing n objects, MinPts and epsilon Eps.**
2. **- Put an edge between each pair of core points within distance Eps of each other.**
3. **- Mark noise points.**
4. **- Make each group of connected core points into a separate cluster.**
5. **- Assign each border point arbitrarily to one of the clusters containing its associated core pts.**

Output: Set of clustered objects.

3) HDBSCAN: Is a variant of the DBSCAN cluster algorithm that focuses on finding hierarchical results. In HDBSCAN, cutoff points are left behind and only the center points of the group are considered. The number of clusters formed by HDBSCAN that analyzes one million data is 586 clusters. Like DBSCAN, HDBSCAN is also a dynamic grouping technique. Regular DBSCAN is great for grouping data in different ways, but it falls short of grouping data with different densities. HDBSCAN is based on the DBSCAN algorithm, but it actually works with different epsilon values, so it only requires a single parameter, that is, the minimum cluster size.

H. System Architecture

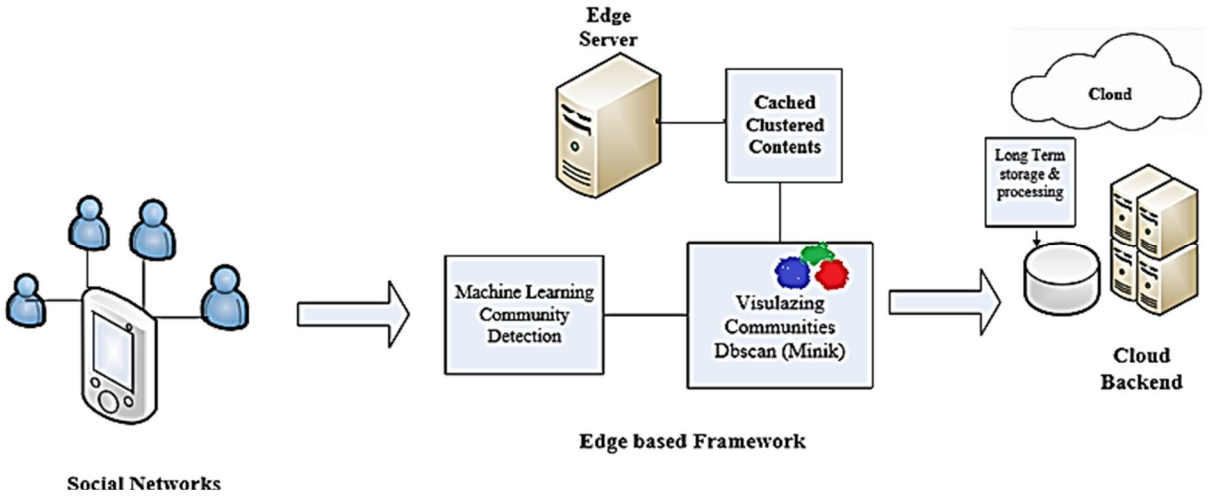


Fig. 4.1 System Architecture

I. Simulation Framework

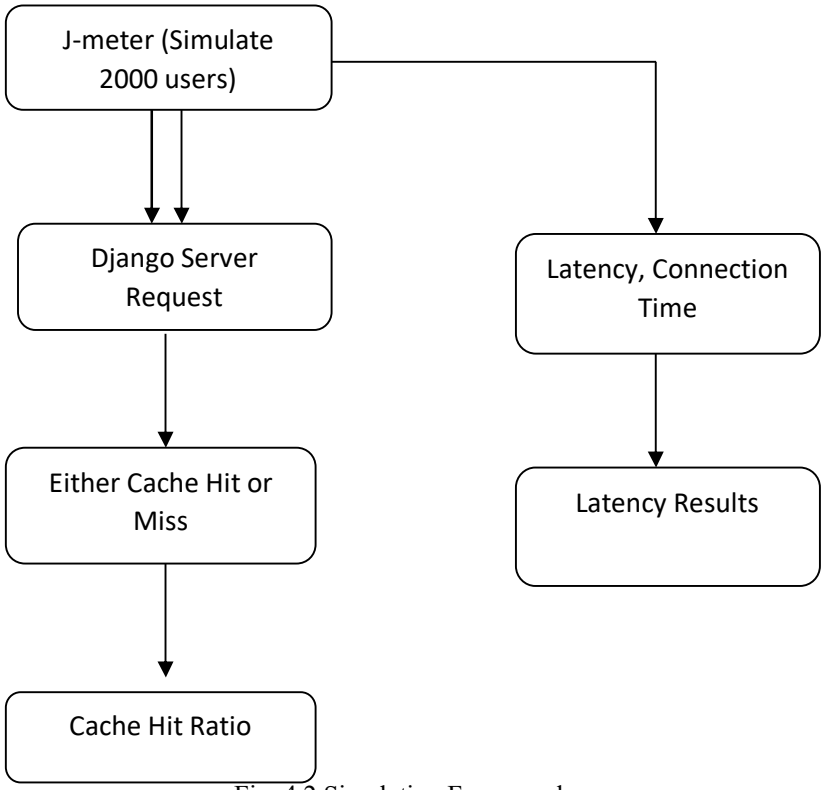
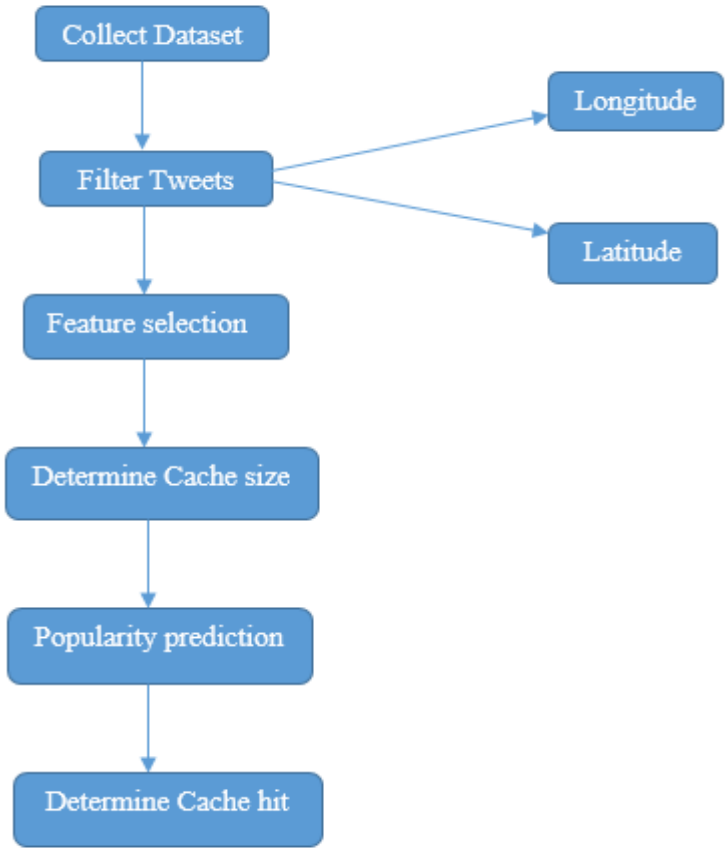


Fig. 4.2 Simulation Framework

III. FLOWCHART OF PROPOSED FRAMEWORK



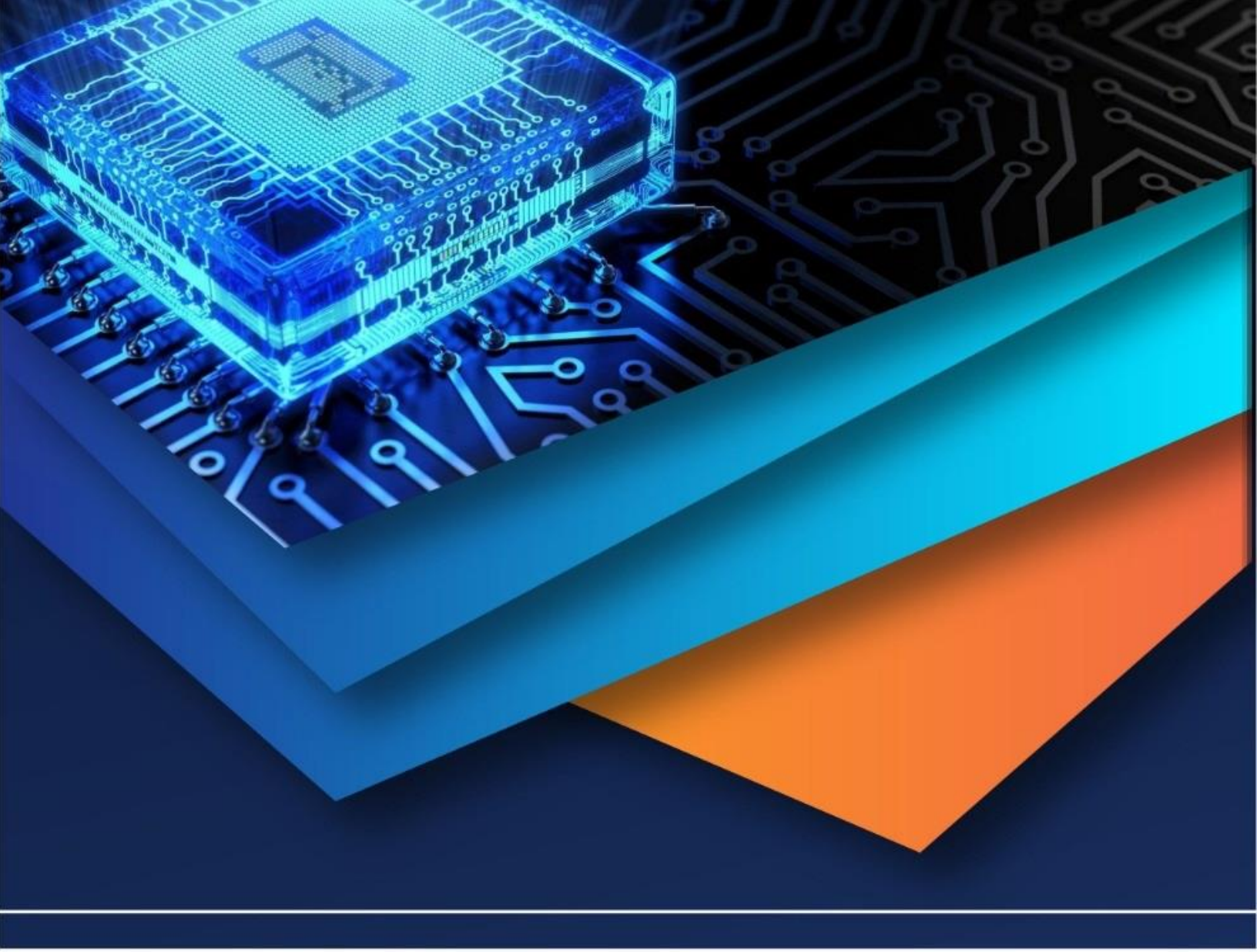
IV. CONCLUSION

In our work we proposed an edge based caching policy. To elevate core network load and minimize user latency, response time and increase cache hit ratio, we first find similarity in the dataset and then find popular tweets from the data using some parameters. Using python Django DB we cached the popular tweets from the data near the edge. So using caching technique in the network minimize user delay time as well as response time and increase cache hit rate in the network.

REFERENCES

[1] Q. Ding, H. Pang, and L. Sun, "SAM: Cache space allocation in collaborative edge-caching network," IEEE Int. Conf. Commun., no. May, 2017.
[2] L. Li, G. Zhao, S. Member, and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks : Research Issues and Challenges in Content Placement and Delivery Strategies," no. c, pp. 1–48, 2018.
[3] R. Haw, S. M. A. Kazmi, K. Thar, M. G. R. Alam, and C. S. Hong, "Cache aware user association for wireless heterogeneous networks," IEEE Access, vol. 7, pp. 3472–3485, 2019.
[4] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," Proc. - IEEE INFOCOM, vol. 2016-July, 2016.
[5] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," 2018 52nd Annu. Conf. Inf. Sci. Syst. CISS 2018, pp. 1–6, 2018
[6] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive Content Caching by Exploiting Transfer Learning for Mobile Edge Computing," 2017 IEEE Glob. Commun. Conf. GLOBECOM 2017 - Proc., vol. 2018-Janua, no. June 2017, pp. 1–6, 2018.
[7] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward Smart and Cooperative Edge Caching for 5G Networks : A Deep Learning Based Approach."
[8] F. Miao, D. Chen, and L. Jin, "Multi-level PLRU Cache Algorithm for Content Delivery Networks," 2017.
[9] Y. N. B and T. Shigeyasu, for Reducing Network Traffic and Improving Cache Hit Rate on NDN. Springer International Publishing, 2019.
[10] M. Lee, F. Leu, and Y. Chen, "Cache Replacement Algorithms for YouTube," 2014
[11] N. Zhang, K. Zheng, and M. Tao, "Using Grouped Linear Prediction and Accelerated Reinforcement Learning for Online Content Caching."
[12] P. Bedi, and C. Sharma, "Community detection in social networks," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 6, no. 3, pp. 115-135, 2016.

[13] D. Sculley, "Web-scale k-means clustering." pp. 1177-1178.
[14] D.-T. Huynh, X. Wang, T. Q. Duong, N.-S. Vo, and M. Chen, "Social-aware energy efficiency optimization for device-to-device communications in 5G networks," Computer Communications, vol. 120, pp. 102-111, 2018
[15] Y. M. ElBarawy, R. F. Mohamed, and N. I. Ghali, "Improving social network community detection using DBSCAN algorithm." pp. 1-6.
[16] Q. Ding, H. Pang, and L. Sun, "SAM: Cache space allocation in collaborative edge-caching network," IEEE Int. Conf. Commun., no. May, 2017.
[17] L. Li, G. Zhao, S. Member, and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks : Research Issues and Challenges in Content Placement and Delivery Strategies," no. c, pp. 1–48, 2018.
[18] R. Haw, S. M. A. Kazmi, K. Thar, M. G. R. Alam, and C. S. Hong, "Cache aware user association for wireless heterogeneous networks," IEEE Access, vol. 7, pp. 3472–3485, 2019.
[19] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," Proc. - IEEE INFOCOM, vol. 2016-July, 2016.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY