

# Learning-based Caching in Cloud-Aided Wireless Networks

Syed Tamoor-ul-Hassan\*, Sumudu Samarakoon\*, Mehdi Bennis\*, Matti Latva-aho\*, Choong-Seong Hong†

\*Center for Wireless Communications, University of Oulu, Finland,

email: {tsyed, bennis, sumudu, matti.latva-aho}@ee.oulu.fi

†Department of Computer Engineering, Kyung Hee University, South Korea, email: cshong@khu.ac.kr

**Abstract**—This paper studies content caching in cloud-aided wireless networks where small cell base stations with limited storage are connected to the cloud via limited capacity fronthaul links. By formulating a utility (inverse of service delay) maximization problem, we propose a cache update algorithm based on spatio-temporal traffic demands. To account for the large number of contents, we propose a content clustering algorithm to group similar contents. Subsequently, with the aid of regret learning at small cell base stations and the cloud, each base station caches contents based on the learned content popularity subject to its storage constraints. The performance of the proposed caching algorithm is evaluated for sparse and dense environments while investigating the tradeoff between global and local class popularity. Simulation results show 15% and 40% gains in the proposed method compared to various baselines.

## I. INTRODUCTION

Edge caching represents a viable solution to overcome challenges associated with network densification by intelligently caching contents at the network edge [1]. Besides reducing latency, edge caching also offloads the backhaul traffic load [2]. Existing literature investigates the potential benefits of caching in terms of backhaul offloading gains and latency [3]–[5]. While these works show improved network performance through caching, they neglect the intrinsic user behavior by considering a fixed caching policy. Due to the spatio-temporal requests, small cell base stations (SBSs) often need to update their cache following their local request distribution to minimize latency [6]. In such scenarios, optimal content placement becomes a challenging and non-trivial problem. To serve user requests, the works in [7]–[9] proposed dynamic caching algorithms based on fixed content popularity. However, these works assume a small content library with fixed content popularities. With the growing library size, determining popularity and content caching becomes computationally expensive. Recently, grouping contents based on their popularity was proposed in [10]. However, how to group contents and cache accordingly based on time-varying popularity was not studied.

The main contribution of this paper is to revisit the fundamental problem of content caching under spatio-temporal traffic demands in cloud-aided wireless networks and explore the tradeoffs between global and local content popularity. By considering a random deployment of SBSs and users, the objective is to determine what contents need to be cached locally by every SBS so as to maximize the cache hit rate. Based on the instantaneous content requests, each SBS locally learns the time-varying content popularity with the aid of regret learning [11]. Simultaneously, the cloud learns the global content popularity. By randomizing its caching strategy, each SBS optimizes the caching policy in a decentralized manner and updates its cache.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

Consider the downlink transmission of a small cell network comprised of randomly deployed SBSs,  $\mathcal{S} = \{1, \dots, S\}$  with intensity  $\lambda_{\text{SBS}}$ . Let  $Y_s$  represent the location of the  $s$ -th SBS. Each SBS serves a set of user equipment(s) (UEs),  $\mathcal{U} = \{1, \dots, U\}$ , deployed randomly with intensity  $\lambda_{\text{UE}}$ . The location of the  $u$ -th UE is denoted by  $Z_u$ . Each SBS serves UEs' requests over a common pool of spectrum with bandwidth  $\omega$ . Accordingly, the instantaneous data rate of UE  $u$  served by SBS  $s$  is:

$$R_{su}(t) = \omega \log_2 \left( 1 + \frac{p_s \|\Psi_{su}(t)\|^2}{\sigma^2 + \sum_{s' \in \mathcal{S} \setminus s} p_{s'} \|\Psi_{s'u}(t)\|^2} \right), \quad (1)$$

where  $\sigma^2$  represents the variance of noise,  $p_s$  denotes the transmit power of SBS  $s$  and  $\Psi_{su}(t)$  denotes the channel gain between UE  $u$  and SBS  $s$ .

Each SBS is equipped with a cache of size  $d$  where it stores contents from a content library  $\mathcal{F} = \{1, \dots, F\}$  as shown in Fig. 1. Let  $1/\mu$  be the size of all contents. In addition, let  $\Xi(t) = [\Xi_s(t)]_{s \in \mathcal{S}}$  represent the vector of SBSs cache at time  $t$  where  $\Xi_s(t) \subseteq \mathcal{F}$  represents the contents cached by SBS  $s \in \mathcal{S}$  at time  $t$  such that  $|\Xi_s(t)| \leq d$ . We assume that SBSs partition the content library into popularity classes such that each content in a class is equally popular i.e., *multi-class model* [10]. Let the set of contents be partitioned into popularity classes  $\mathcal{K} = \{1, 2, \dots, K\}$ , where  $\mathcal{F}_k = \{1, \dots, F_k\}$  such that  $\mathcal{F}_k \subset \mathcal{F}$ ,  $\mathcal{F}_k \cap \mathcal{F}_{k'} = \emptyset$ ,  $k \neq k'$ . Due to the constrained cache size and lack of coordination among SBSs, each SBS is connected to the cloud via a fixed capacity fronthaul link  $C_f$  to obtain the global content popularity and update its cache accordingly.

Each UE requests contents from the library following the dynamic popularity model i.e., spatio-temporal model [12]. Let the content demanded by the  $u$ -th UE at time  $t$  is denoted by  $q_u(t)$  such that  $q_u(t) \in \{0, 1, 2, \dots, F\}$  where  $q_u(t) = 0$  denotes no request by user  $u$  at time  $t$ . For simplicity, we assume that each UE requests one content at a time. Let the content demand vector at SBS  $s$  be  $\mathbf{D}_s(t) = [D_{sf}(t)]_{f \in \mathcal{F}}$  such that  $D_{sf}(t) = \sum_{u \in \mathcal{N}_s} \mathbb{1}_{q_u(t)=f}$  where  $\mathbb{1}_x$  is the indicator function and  $\mathcal{N}_s$  denotes the users in the coverage of SBS  $s$ .

The instantaneous reward of a SBS depends on the instantaneous cache hits and service rate. Absence of a requested content from a SBS incurs a cache miss. If the content is cached by multiple SBSs in UE's coverage, the user associates to the nearest SBS caching the requested content. In this regard, the reward of SBS  $s$  for serving UE  $u$  is given by:

$$g_{sq_u}(t, \Xi_s(t)) = \mathbb{1}_{\{q_u(t) \in \Xi_s(t)\}} R_{su}(t). \quad (2)$$

### A. Utility Maximization Problem

The objective of SBSs is to determine a caching policy that maximizes their reward while ensuring UEs QoS. From (2),

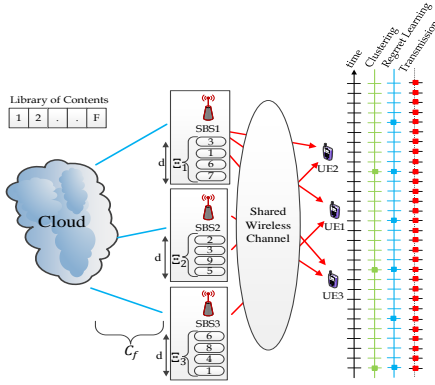


Fig. 1. System Model

it can be observed that the reward of a SBS depends on the achievable rate and caching policy, i.e., SBS is rewarded if and only if it caches the requested content. For simplicity, fronthaul links are assumed to be used only for cache update and service rate is considered to be zero if the SBS has not cached the content. One of the challenges associated with cache update is when the number of most popular contents is larger than the cache size. In this case, SBSs must update their caching decisions carefully as caching less popular contents may decrease the SBS's reward.

For a few UE requests, content popularity at SBSs may not be determined accurately, resulting in poor caching policy yielding lower reward. Hence, it is important that enough statistics are available to better learn the content popularity. To overcome this issue, the cloud estimates the global content popularity gathered from all SBSs. However, acquiring global demand and cache update incurs additional cost given by:

$$\varepsilon_s = 1 - \frac{\tau_s}{T_2} \quad (3)$$

where  $\tau_s < T_2$  is the time required for cache update and  $T_2$  represents the time during which the users' requests are observed. Assume  $C_s$  is the fronthaul capacity for SBS  $s$ , the time required to update the cache of SBS  $s$  is:

$$\tau_s = l_p \frac{N}{\mu C_s}, \quad (4)$$

where  $l_p > 0$  is a constant and  $N$  represents the number of new contents. Let  $\Xi_s$  be the vector of caching policies at SBS  $s$  over time  $t = \{0, 1, 2, \dots\}$ , i.e.  $\Xi_s = [\Xi(0), \Xi(1), \Xi(2), \dots]$  and  $\bar{g}_{su}(\Xi_s) = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} g_{su}(\tau, \Xi_s(\tau))$  be the limiting time-average reward at SBS  $s$ . Then, for each SBS  $s \in \mathcal{S}$ , the average per-SBS utility is the aggregate utility of the associated UEs i.e.,

$$\Upsilon_s(\Xi_s) = \varepsilon_s \sum_{\forall u \in \mathcal{N}_s} \Upsilon_{su}(\bar{g}_{su}(\Xi_s)), \quad (5)$$

where  $\Upsilon_{su}(\bar{g}_{su}) = (1/\mu)^{-1} \bar{g}_{su}$ . The network utility maximization problem is:

$$\underset{\Xi_s}{\text{maximize}} \quad \sum_{\forall s \in \mathcal{S}} \Upsilon_s(\Xi_s) \quad (6a)$$

$$\text{subject to} \quad |\Xi_s| \leq d, \quad \forall s \in \mathcal{S} \quad (6b)$$

$$g_{squ}(t) > g_{\min}, \quad \forall u \in \mathcal{U}, \forall t \quad (6c)$$

$$\sum_{\forall s \in \mathcal{S}} C_s \leq C_f, \quad (6d)$$

$$|Y_{s(u)} - Z_u| < |Y_{s'} - Z_u|, \quad \forall u \in \mathcal{U}, s' \in \mathcal{S} \setminus s(u), \quad (6e)$$

where (6c) is the minimum QoS threshold, (6e) represents a nearest UE-SBS association, (6d) is the fronthaul capacity constraint and  $s(u)$  represents the serving SBS of user  $u$ .

### III. DEMAND-BASED CONTENT CLUSTERING

The problem in (6) is trivial when  $F \approx d$ . With the increasing library size, the problem becomes non-trivial. Moreover, due to time-varying popularity of contents, the complexity of (6) increases manifolds, making the problem extremely challenging to solve. It has been observed that in a real system, there exists a correlation among contents requests i.e., request of a content is nearly similar to one or more contents [4]. This suggests grouping contents based on their demands as a solution to improve caching decisions. By observing the content demand over a finite time period, contents are clustered into different classes where contents in the same class have similar popularity. Thus, (6) is solved over classes rather than contents. In this work, a similarity measure between demand vectors is used to cluster contents into classes. Since, content similarity varies slowly over time, content clustering is a slower process than cache update. In other words, the content clustering remains fixed for a period  $T_1 > T_2$  where  $T_2$  is the cache update time. To define the similarity measure, let  $\mathbf{M}(t) = [M_{ff'}(t)]_{f, f' \in \mathcal{F}}$  be the similarity matrix at time  $t$  with:

$$M_{ff'}(t) = \exp \left( -\frac{|D_f(t)\pi_f(t) - D_{f'}(t)\pi_{f'}(t)|^2}{2\sigma_f^2} \right) \quad \forall f' \in \mathcal{F}, \quad (7)$$

where  $\pi_f(t)$  is the popularity of file  $f$  at time  $t$ ,  $D_f(t)$  is the instantaneous request of file  $f$  and  $\sigma_f^2$  controls the impact of popularity on similarity. In order to find the content demand vector over the network, all the SBSs upload their demand vectors  $\mathbf{D}_s(t)$  to the cloud which computes the network wide demand vector  $\mathbf{D}'(t) = \sum_{s \in \mathcal{S}} \mathbf{D}_s(t)$  and broadcasts it to all SBSs. Thereafter, SBSs perform content clustering based on the following demand vector:

$$\mathbf{D}_s'(t) = \alpha \mathbf{D}'(t) + (1 - \alpha) \mathbf{D}_s(t) \quad 0 \leq \alpha \leq 1, \quad (8)$$

where  $\alpha$  is a tunable parameter that captures local vs. global demand. In this work, spectral clustering technique is used to perform content clustering [13] that exploits the frequency of content requests from the users in the coverage of SBSs and the variance of the similarity matrix to form content classes. The content clustering algorithm at each SBS is explained in Algorithm 1.

### IV. CACHING VIA REINFORCEMENT LEARNING

The main objective of an efficient caching strategy is to maximize the cache hits while minimizing the service delay and fronthaul cost. However, designing an efficient caching strategy is extremely challenging without a prior knowledge of user demands. Since the demand vector at each SBS varies from other SBSs due to their spatial location, it is necessary to devise adaptive decentralized algorithms to determine the caching strategy. In this respect, each SBS leverages reinforcement learning (RL) to accurately estimate the caching strategy that maximizes the payoff.

To employ RL, each SBS implicitly learns the class popularity based on instantaneous user demands. As per (6b), the SBSs cache a subset of library contents. At each time, the SBS determines the set of library content to cache which defines the actions of SBSs. Hence, the action space comprises of caching content/contents of class/classes. Let  $\mathcal{A}_s$  denotes the

---

**Algorithm 1: Content Clustering and Cache Update**


---

**Input:** Observed local content demand vector  $D_s(t)$  and Global/local tradeoff parameter  $\beta$ .

**Result:** Content cluster at SBSs  $\mathcal{K}_s = \{1, \dots, K_s\}$ ,  $\forall s \in \mathcal{S}$ .

**Algorithm:**

**Phase I - Similarity Matrix Computation;**

- Transmit the local demand vector  $D_s(t)$  to the cloud.
- Compute the similarity matrix  $M(t)$  based on (7).

**Phase II - Spectral Clustering Algorithm;**

- Compute the diagonal degree matrix  $X$  where  $X_i = \sum_{\forall f \in \mathcal{F}} m_{ij}$ .
- Compute the graph laplacian matrix  $L = X - M(t)$ .
- Normalize the graph laplacian matrix  $L_{\text{norm}} = X^{-\frac{1}{2}} L X^{\frac{1}{2}}$ .
- Select a number of  $k_{\text{max}}$  eigenvalues of  $L_{\text{norm}}$  such that  $\lambda_1 \leq \dots \leq \lambda_{i_{\text{max}}}$  where  $k_{\text{max}}$  is the maximum number of clusters and  $\lambda_i$  is the  $i$ -th smallest value of  $L$ .
- Choose  $k = \max_{i=k_{\text{min}}, \dots, k_{\text{max}}} \Delta_i$  where  $\Delta_i = \lambda_{i+1} - \lambda_i$ .
- Calculate  $k$  smallest eigenvectors and apply  $k$ -means clustering to cluster rows of eigenvectors.

**Phase III - Regret Learning and Cache Update;**

- Each SBS learns the probability distribution vector  $\pi_s$  based on (11).
  - The cloud learns the probability distribution vector  $\pi_c$  based on (11).
  - Each SBS updates its cache based on the mixed distribution  $\pi' = (1 - \beta)\pi_s + \beta\pi_c$ .
- 

action space of SBS  $s$  where  $\mathcal{A}_s = [\Xi_s^{k_s}]_{k_s \in \mathcal{K}_s}$  where  $\mathcal{K}_s$  represents the set of popularity classes at SBS  $s$ . Here, the action  $\Xi_s^{k_s} = 1$  indicates that SBS  $s$  caches content(s) of class  $k_s$ . Thus (5) can be rewritten as:

$$\Upsilon_s(\Xi_s^{k_s}) = \varepsilon_s \sum_{\forall u \in \mathcal{S}(u)} \Upsilon_{su}(\Xi_s^{k_s}). \quad (9)$$

Since, the requests of users change over time, it is necessary to adapt the caching strategy accordingly. As a result, the caching decision corresponding to content(s) of a class becomes a random variable. Let the probability distribution of the caching strategy at SBS  $s$  be  $\pi_s(t) = [\pi_{s, \Xi_s^1}(t), \dots, \pi_{s, \Xi_s^{k_s}}(t)]$  where  $\pi_{s, \Xi_s^{k_s}}(t) = \mathbb{P}(\Xi_s(t) = \Xi_s^{k_s})$  such that  $\sum_{\Xi_s^{k_s} \in \mathcal{A}_s} \pi_{s, \Xi_s^{k_s}}(t) = 1$ .

Let  $\tilde{\Upsilon}_s(t) = (\tilde{\Upsilon}_{s, \Xi_s^1}(t), \dots, \tilde{\Upsilon}_{s, \Xi_s^{k_s}}(t))$  denote the vector of estimated utilities for all actions of SBS  $s$  where  $\tilde{\Upsilon}_{s, \Xi_s^{k_s}}(t)$  is the estimated utility for action  $\Xi_s^{k_s}$  at time  $t$ . Further, let  $\hat{\Upsilon}_s(t)$  be the feedback of the utilities from all associated users. Due to the time-varying content demands, each SBS needs to update its cache to maximize the utility. For this, each SBS uses regret learning mechanism to determine the caching strategy. The regret learning mechanism iteratively allows players to explore all possible actions and learn optimal strategies [11]. As a result, the main objective of utility maximization recast as a regret minimization problem. Here, the objective is to exploit the actions that yield higher rewards while exploring other actions with lower regrets. This behavior is captured by the Boltzmann-Gibbs (BG) distribution given as [7]:

$$G_{s, \Xi_s^{k_s}}(\tilde{r}_s(t)) = \frac{\exp(\frac{1}{\xi_s} \tilde{r}_{s, \Xi_s^{k_s}}^+(t))}{\sum_{\forall \Xi_s^{k_s} \in \mathcal{A}_s} \exp(\frac{1}{\xi_s} \tilde{r}_{s, \Xi_s^{k_s}}^+(t))}, \quad \forall \Xi_s^{k_s} \in \mathcal{A}_s, \quad (10)$$

where  $\xi_s > 0$  is a temperature coefficient, and  $\tilde{r}_{s, \Xi_s^{k_s}}^+(t) = \max(0, \tilde{r}_{s, \Xi_s^{k_s}}(t))$ . A small value of  $\xi_s$  maximizes the sum of regrets which results in a mixed strategy where SBSs exploits the actions with higher regrets at time period  $t$ . On the contrary, a higher value of  $\xi_s$  results in uniform distribution

over the action set. At each time instant, the estimation of the utility, regret and probability distribution over the action space  $\mathcal{A}_s, \forall s \in \mathcal{S}$  is given as:

$$\begin{aligned} \tilde{\Upsilon}_{s, \Xi_s^{k_s}}(t) &= \tilde{\Upsilon}_{s, \Xi_s^{k_s}}(t-1) + \Gamma_s^1(t) \mathbb{1}_{\{\Xi_s(t) = \Xi_s^{k_s}\}} \left[ \hat{\Upsilon}_s(t) - \tilde{\Upsilon}_{s, \Xi_s^{k_s}}(t-1) \right] \\ \tilde{r}_{s, \Xi_s^{k_s}}(t) &= \tilde{r}_{s, \Xi_s^{k_s}}(t-1) + \Gamma_s^2(t) \left( \tilde{\Upsilon}_{s, \Xi_s^{k_s}}(t) - \hat{\Upsilon}_s(t) - \right. \\ &\quad \left. \tilde{r}_{s, \Xi_s^{k_s}}(t-1) \right) \end{aligned} \quad (11)$$

$$\pi_{s, \Xi_s^{k_s}}(t) = \pi_{s, \Xi_s^{k_s}}(t-1) + \Gamma_s^3(t) \left( G_{s, \Xi_s^{k_s}}(\tilde{r}_s(t)) - \pi_{s, \Xi_s^{k_s}}(t-1) \right),$$

where the learning rates  $\Gamma_s^i(t) \forall i \in \{1, 2, 3\}$  satisfy [11]:

$$\begin{aligned} (i) \quad &\lim_{t \rightarrow \infty} \sum_{\tau=1}^t \Gamma_s^i(\tau) = +\infty, \quad \lim_{t \rightarrow \infty} \sum_{\tau=1}^t (\Gamma_s^i(\tau))^2 < +\infty \\ (ii) \quad &\lim_{t \rightarrow \infty} \sum_{\tau=1}^t \frac{\Gamma_s^2(t)}{\Gamma_s^1(t)} = 0, \quad \lim_{t \rightarrow \infty} \sum_{\tau=1}^t \frac{\Gamma_s^3(t)}{\Gamma_s^2(t)} = 0. \end{aligned}$$

Unlike SBSs, the cloud has the knowledge on the demands over the whole network. Based on this global knowledge, cloud learns the caching strategy  $\pi_c$  using the steps of (11) by modifying the action vector to  $\mathcal{A}_c = [\mathcal{A}_s]_{s \in \mathcal{S}}$ , the corresponding utilities to  $\Upsilon_c(\mathcal{A}_c) = \sum_{s=1}^S \Upsilon_s(\Xi_s^{k_s}, \bar{g}_s)$  and regret estimations to  $\tilde{r}_c(\mathcal{A}_c) = \sum_{s=1}^S \tilde{r}_{s, \Xi_s^{k_s}}$ .

**A. Cache Eviction Algorithm**

To update the SBSs cache, existing contents need to be evicted due to the constrained cache size. For simplicity, we assume only a single content is evicted at time  $t$ . At every time  $T_2$ , each SBS observes the request for the cached contents. Based on the number of requests, each SBS builds the Gibbs-Sampling based distribution as:

$$G_{sf}(t) = \frac{\exp(-\sum_{\tau=1}^{t-1} \pi_{sf}(\tau))}{\sum_{\forall f' \in \Xi_s} \exp(-\sum_{\tau=1}^{t-1} \pi_{sf'}(\tau))}, \quad \forall f \in \Xi_s. \quad (12)$$

From the above equation, the content with least popularity will be evicted from the cache. Using the Gibbs-Sampling based probability distribution, each SBS evicts the content and caches new content based on  $\pi'$  given by:

$$\pi' = (1 - \beta)\pi_s + \beta\pi_c, \quad (13)$$

where  $\pi_s$  and  $\pi_c$  represents the caching strategy at SBS  $s$  and cloud respectively and  $\beta$  captures the local/global tradeoff. Note that due to the assumption of time scale separation over three phases therein, the proposed solution does not assure global optimality of the network utility maximization.

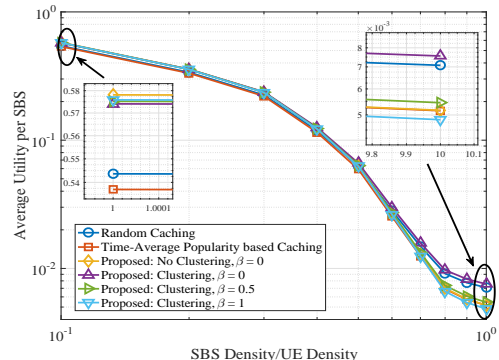


Fig. 2. Average utility per SBS vs SBS Density/UE Density,  $p_s = 23\text{dBm}$ ,  $d = 50$ ,  $F = 500$ ,  $\xi_c = 0.0002$ ,  $\xi_s = 0.01$ ,  $\alpha = \beta$

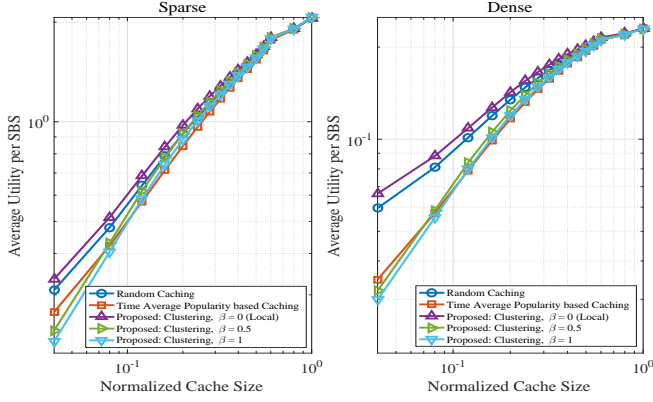


Fig. 3. Average Utility vs cache size,  $C_f = 50\text{Gbps}$ ,  $\alpha = \beta$

## V. SIMULATION RESULTS

In this section, we analyze the performance of the proposed mechanism and examine insights of the local/global tradeoff ( $\beta$ ) under several deployment and caching scenarios. By assuming a system bandwidth of 1.4MHz, the performance of the proposed scheme is compared against two baseline schemes: random caching (B1) and time-average content popularity based caching (B2). Both baselines and proposed solution uses random RBs to serve users' requests. Further,  $\frac{\lambda_{\text{SBS}}}{\lambda_{\text{UE}}} = 0.1$  denotes a sparse network while  $\frac{\lambda_{\text{SBS}}}{\lambda_{\text{UE}}} = 1$  denotes a dense network. Fig. 2 shows the per-SBS utility as a function of the ratio of SBS density to user density. With increased  $\lambda_{\text{SBS}}/\lambda_{\text{UE}}$ , the gains of the proposed scheme ( $\beta = 0$ ) vary from 6%-10% and 8%-40% compared to B1 and B2, respectively. Meanwhile, the proposed scheme with clustering ( $\alpha = \{0, 0.5\}$ ) achieves 23%, 6% gains over the proposed scheme without clustering.

Fig. 3 shows the variation of the per-SBS utility as a function of cache size. For a small cache size, the proposed scheme ( $\beta = 0$ ) achieves {10%, 13%} and {25%, 56%} gains over baselines B1 and B2, respectively for {sparse, dense} scenarios. With the increasing cache size, the proposed scheme ( $\beta = 0$ ) achieves 7% and 28% gains over baselines B1 and B2 for both scenarios.

Fig. 4 and 5 shows the tradeoff between local and global learning. It can be observed that the local clustering always performs better than no clustering approach for sparse and dense scenarios. At  $\beta = 0.8$  for dense scenario, both schemes yield the same utility for small cache size. Further increasing  $\beta$  makes the no clustering approach better than the local clustering. In addition, decreasing the fronthaul capacity has no impact of local/global tradeoff parameter. When the cache size increases, clustering approach is slightly better than non-

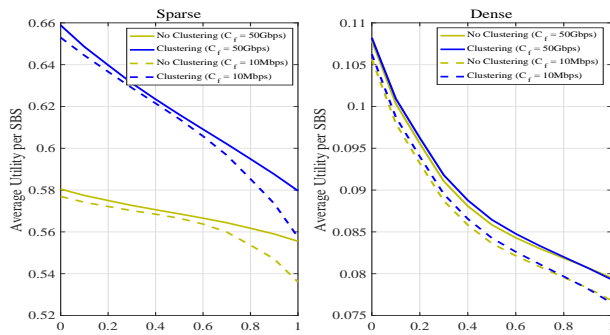


Fig. 4. Local/global tradeoff for sparse/dense scenarios,  $d = 50$ ,  $F = 500$

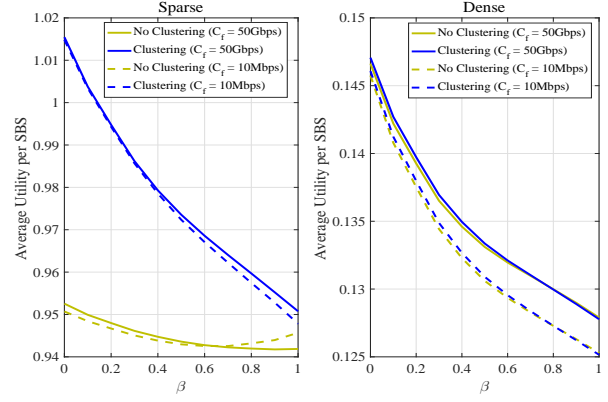


Fig. 5. Local/global tradeoff for sparse/dense scenarios,  $d = 100$ ,  $F = 500$  clustering for dense scenario as shown in Fig. 5. Furthermore, at  $\beta = 0.7$  for dense scenario, both schemes yield the utility. By increasing  $\beta$  further makes the no clustering approach better than the local clustering. Furthermore, there is no impact of fronthaul capacity on  $\beta$ .

## VI. CONCLUSION

In this letter, we investigated content caching in cloud-aided wireless networks, where SBSs store contents from a large content library. We proposed a clustering algorithm based on Gaussian similarity. Using the regret learning mechanism at the SBSs and the cloud, we proposed a per-SBS caching strategy that minimizes the service delay in serving users' requests. In addition, we investigated the tradeoff between local and global content popularity on the proposed algorithm for sparse and dense deployments.

## REFERENCES

- [1] F. Boccardi *et al.*, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74 – 80, Feb. 2014.
- [2] 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, pp. 82 – 89, Aug. 2014.
- [3] K. Shanmugam *et al.*, "FemtoCaching: wireless video content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402 – 8413, Dec. 2013.
- [4] E. Bastug, M. Bennis and M. Debbah, "Cache-enabled small cell networks: modeling and tradeoffs," *11th International Symposium on Wireless Communication Systems (ISWCS)*, Barcelona, Spain, Aug 2014.
- [5] B. Chen, C. Yang and A. F. Molisch, "Cache-enabled device-to-device communications: offloading gain and energy cost," <https://arxiv.org/abs/1606.02866>.
- [6] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018 – 1021, 2010.
- [7] M. S. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Content-aware user clustering and caching in wireless small cell networks," *11th International Symposium on Wireless Communication Systems (ISWCS)*, Barcelona, Spain, Aug 2014.
- [8] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," *IEEE International Conference on Communications (ICC)*, Sydney, Australia, June 2014.
- [9] Z. Chen and M. Kountouris, "Cache-enabled small cell networks with local user interest correlation," *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Stockholm, Sweden, June 2015.
- [10] M. Leconte, M. Lelarge and L. Massoulié, "Bipartite graph structures for efficient balancing of heterogeneous load," in *Proc. SIGMETRICS*, NewYork, USA, June, 2012.
- [11] M. Bennis, S. M. Perlaza and M. Debbah, "Learning coarse correlated equilibrium in two-tier wireless networks," in *Proc. IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 2012.
- [12] M. Leconte *et al.*, "Placing Dynamic Content in Caches with Small Population," *IEEE INFOCOM*, 2016.
- [13] U. von Luxburg, A tutorial on spectral clustering, *Stat. Comput.* 17 (4) (2007) 395 – 416.