A Deep Learning Approach for Optimizing Content Delivering in Cache-Enabled HetNet

Lei Lei[1], Lei You[2], Gaoyang Dai[2], Thang Xuan Vu[1], Di Yuan[2], and Symeon Chatzinotas[1]

[1]Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[2]Department of Information Technology, Uppsala University, Sweden
Emails: {*lei.lei; thang.vu; symeon.chatzinotas@uni.lu; lei.you; di.yuan; gaoyang.dai@it.uu.se*}

***Abstract*—In ultra-dense heterogeneous networks, caching popular contents at small base stations is considered as an effective way to reduce latency and redundant data transmission. Optimization of caching placement/replacement and content delivering can be computationally heavy, especially for large-scale networks. The provision of both time-efficient and high-quality solutions is challenging. Conventional iterative optimization methods, either optimal or heuristic, typically require a large number of iterations to achieve satisfactory performance, and result in considerable computational delay. This may limit their applications in practical network operations where online decisions have to be made. In this paper, we provide a viable alternative to the conventional methods for caching optimization, from a deep learning perspective. The idea is to train the optimization algorithms through a deep neural network (DNN) in advance, instead of directly applying them in real-time caching or scheduling. This allows significant complexity reduction in the delay-sensitive operation phase since the computational burden is shifted to the DDN training phase. Numerical results demonstrate that the DNN is of high computational efficiency. By training the designed DNN over a massive number of instances, the solution quality of the energy-efficient content delivering can be progressively approximated to around 90% of the optimum.**

***Index Terms*—Deep neural network, caching, energy optimization, user clustering, heterogeneous network.**

I. Introduction

Deploying cache-enabled small base stations (SBSs) in ultra-dense heterogeneous networks (HetNets) has been considered as one of the promising solutions to meet the high performance requirements in the fifth generation (5G) systems [1]. Caching the files with high popularity to SBSs and directly serving most of the users' content requests from the SBSs' local caches, are beneficial towards reducing latency and network traffic load. In the cache-enabled heterogeneous network, optimizing caching placement, user association, and content delivering are the key aspects in performance improvement [2]–[4]. In [2], the authors applied a primal-dual decomposition and subgradient method to solve a cooperative content caching placement problem. In [3], the authors formulated a throughput maximization problem of jointly caching, routing, and scheduling using linear programming. Then a column generation algorithm was applied to solve the problem. In [4], the authors characterized the complexity in solving a caching placement problem, and developed an approximation algorithm to reduce the content server's load. The majority of these methods in the literature is either tailored exact algorithms or meticulously designed heuristics. In these iterative algorithms, performance improvement may typically lead to higher computational complexity and algorithm running time.

The practical network has to make online decisions in real-time operations, e.g., content caching, scheduling, and resource allocation. Then the network scheduler may need to make a trade-off, that is, tuning down the algorithm complexity (or apply simple heuristics) to meet strict delay requirements, e.g., millisecond scheduling period in LTE systems [5]. This typically degrades the network performance. Thus for practical operations, it is important to deliver both computational-efficient and high-quality solutions. Machine learning, as an emerging approach in the toolset for wireless communications, has received considerable research attention in recent years [6]. In addition to conventional heuristics, it provides another viable choice in developing efficient solutions. In [7], the authors applied a reinforcement learning approach for solving a classical job scheduling problem. In [8], the authors considered a Q-learning based method to minimize the total transmission cost in caching optimization.

In this paper, we aim at minimizing network energy consumption and reducing transmission delay via optimizing caching placement, user association, and content delivering. We employ a deep neural network (DNN) to train the optimal scheduling algorithm through the DNN to make it learn to optimize in the training phase, such that in the DNN operation phase, time-efficient and near-optimal solutions can be obtained in real-time caching optimization. Specifically, the contributions are summarized below:

- We investigate the energy-efficient caching design by jointly optimizing caching placement, user association, and content delivering. We formulate a joint optimization problem based on the simple transmission strategy, mainly to optimize the user association and the file assignment. Based on these optimized results, we then develop an enhanced content delivering scheme, in order to optimize the content transmission such that all users can obtain the requested files in time, and energy consumption can be reduced. Both problems can be optimally solved by iterative optimization algorithms which may have high computational complexity in general.

978-1-5090-3009-5/17/$31.00

- Towards real-time network operations, we develop a learning-based approach to efficiently deliver the solution for caching optimization. Specifically, we design a DNN framework to train the optimal algorithms through extensive training. Moreover, we optimize the DNN framework based on the specific problems' structure. After the training phase, the output can be obtained by providing the required input parameters to the DNN.
- We carry out simulations to demonstrate the promising performance of the developed deep learning approach. The trained DNN can provide approximative solutions to the optimum. The solution quality, i.e., the performance gap to the optimum, can achieve a satisfactory level after the training stage. The numerical results also demonstrate the DNN's superiority in computational efficiency in its operation phase over iterative optimization algorithms.
- We investigate the performance scalability from a new perspective. Compared to the iterative optimization methods, in our proposed approach, the trade-off between computational efforts and solution quality has been shifted from the real-time operation phase to the DNN training phase, which provides new means to address the trade-off issue.

II. System Model

We consider a two-tier heterogeneous network consisting of an MBS m and N SBSs for serving K mobile users. All the SBSs are within the service area of the MBS. We denote the user and the SBS sets as $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ and $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$, respectively. Each SBS n is equipped with a cache with limited storage capacity C_n. There are $\mathcal{F} = \{1, \ldots, f, \ldots, F\}$ files in the list that can be requested from K users. We use L_f to denote the size (in bits) of file f.

In the cache placement phase, the F files can be selectively stored and updated at each cache upon demand. The MBS is connected to the core network and is always able to serve all the users' requested files. If the requested file is not available in the SBS cache, it can be served directly from the MBS. In practice, users may prefer to download files from the nearby SBS's cache. It may not be wise for some cell-edge users to download the file from the MBS due to poor channel conditions. Note that in this work, the files to be stored at the SBSs and determining which base stations (BSs) to serve users for transmitting their requested files will be an outcome of optimization.

In the content delivery phase, one channel with bandwidth B_n is allocated to SBS n for transmitting data to its associated users, and B_0 is the channel bandwidth for the MBS. Frequency resources are orthogonally allocated among SBSs and the MBS. We consider two schemes in content delivering, time division multiple access (TDMA) and user clustering. The former is conventional. The latter is used to deliver all users' demand by consuming less energy within a time limit. In the user clustering scheme, multiple users served by one BS can be grouped into a set, referring to a cluster [9]. If a cluster is scheduled, all the users in the cluster will be scheduled in the same frequency band to transmit data simultaneously. For example in SBS n, given its associated users in set $\mathcal{K}_n$, a cluster s refers to a group of users, denoted by $\mathcal{K}_n^s$. In total there are $2^{|\mathcal{K}_n|} - 1$ possible clusters for SBS n. These clusters can be selectively and sequentially scheduled in optimization process to deliver users' requested data. Let $s \in \mathcal{S}_n$ be the index of the sth cluster, where $\mathcal{S}_n$ is the set of all candidate clusters for SBS n. Once a cluster $s \in \mathcal{S}_n$ is scheduled in a downlink channel, from SBS n to user k, the signal-to-interference-plus-noise ratio (SINR) for user $k \in \mathcal{K}_n^s$ can be expressed as,

$$\text{SINR}_k^s = \frac{P_{nk}|h_{nk}|^2}{\sum_{j \in \mathcal{K}_n^s \setminus \{k\}} P_{nj}|h_{nk}|^2 + \sigma^2} \tag{1}$$

where P_{nk} is the transmit power of SBS n to user k, and σ^2 is the power of the noise. The channel coefficient of the link between SBS n and user k is denoted by h_{nk}. In addition, let P_{mk} be the transmit power of the MBS to user k, and h_{mk} be the channel coefficient for the link between the MBS and user k. Both h_{mk} and h_{nk} are complex Gaussian random variables with zero mean and unit variance. Note that transmit power P_{nk} for users may not be necessarily uniform, and it can be predefined according to the channel conditions and subject to SBS power constraints.

III. Problem Formulation

In this section, we consider a joint caching problem for content placement, user association, and TDMA-based content delivering. Moreover, according to the outcome of the joint problem, we develop an enhanced scheme to further optimize the content transmission strategy. We aim at optimizing the energy consumption in the network. Next, we formulate the problems in the following.

A. Joint Energy-Efficient Caching Optimization

Given users' file demand $d_{kf} \in \{0, 1\}$ for all k and f, the joint optimization problem amounts to determining which files should be cached in which SBSs, and which SBS should be chosen to transmit a file to a user. We define two sets of binary variables below, and formulate the caching optimization problem in P1.

$$x_{nf} = \begin{cases} 1 & \text{if file } f \text{ is placed at SBS } n\text{'s cache,} \\ 0 & \text{otherwise.} \end{cases}$$

$$z_{nkf} = \begin{cases} 1 & \text{if user } k \text{ is served by } n \text{ to transmit file } f, \\ 0 & \text{otherwise.} \end{cases}$$

Objective (2a) is to minimize the total energy. Power parameters P_{nk} are predefined based on channel conditions and practical power limitations. Parameters T_{nkf} can be precalculated based on TDMA, where $T_{nkf} = L_f / \log_2(1 + \frac{P_{nk}|h_{nk}|^2}{\sigma^2})$. Constraints (2b) confine the storage capacity of each SBS's cache. Constraints (2c) state that a file can be transmitted to a user from an SBS only if this SBS has the file in its cache. In constraints (2d), a file request from a user should be served by only one BS, either an SBS or an MBS, but a user can be

served by more than one SBS to receive requested files. This cooperative transmission can be supported by multi-antenna systems. Constraints (2e) ensure that all users' file requests must be satisfied. Constraints (2f) restrict each SBS' capability in dealing with users' file requests, where A^n_{max} is a maximum number of the files that can be served by SBS n. P1 is a linear integer programming (ILP) problem. The optimal solution can be obtained by exact algorithms [10] or relying on the state-of-the-art solvers. In practice, P1 can be used to identify whether or not the cached contents and the user association need to be adjusted or updated upon the varying instances.

$$\text{P1:} \min_{x_{nf}, z_{nkf}} \sum_{k\in\mathcal{K}} \sum_{n\in\mathcal{N}\cup\{m\}} \sum_{f\in\mathcal{F}} T_{nkf} z_{nkf} P_{nk} \tag{2a}$$

$$\text{s.t.} \sum_{f\in\mathcal{F}} x_{nf} L_f \le C_n,\ \forall n \in \mathcal{N} \tag{2b}$$

$$z_{nkf} \le x_{nf},\ \forall n \in \mathcal{N},\ \forall k \in \mathcal{K},\ \forall f \in \mathcal{F} \tag{2c}$$

$$\sum_{n\in\mathcal{N}\cup\{m\}} z_{nkf} = 1,\ \forall k \in \mathcal{K},\ \forall f \in \mathcal{F} \tag{2d}$$

$$\sum_{n\in\mathcal{N}\cup\{m\}} z_{nkf} \ge d_{kf},\ \forall k \in \mathcal{K},\ \forall f \in \mathcal{F} \tag{2e}$$

$$\sum_{k\in\mathcal{N}} \sum_{f\in\mathcal{F}} z_{nkf} \le A^n_{max},\ \forall n \in \mathcal{N} \tag{2f}$$

B. Enhanced Energy-Efficient Content Delivering

Solving P1 provides the optimized results of file-SBS placement and user-BS association, but adopting TDMA or other simple schemes, e.g., schedule all the users to simultaneously receive files from one SBS, may either lead to high energy consumption or long transmission duration. Based on the outcome of P1, we can further perform optimal cluster scheduling to reduce the transmission delay of TDMA and optimize the whole network energy consumption. We observe that due to orthogonal channel assignment among SBSs and the MBS, once the caching placement and the user association are fixed, the network-level energy-efficient scheduling problem can be decomposed to $N+1$ independent subproblems at the BS level. Each subprobelm is equivalent to solving the following linear programming (LP) problem P2 in cell n, where $n \in \mathcal{N}\cup\{m\}$.

$$\text{P2:} \min_{t_s} \sum_{s\in\mathcal{S}_n} t_s \sum_{k\in\mathcal{K}^s_n} P_{nk} \tag{3a}$$

$$\text{s.t.} \sum_{s\in\mathcal{S}_n} t_s \log_2(1+\text{SINR}^s_k) \ge \sum_{f\in\mathcal{F}_{nk}} L_f,\ \forall k \in \mathcal{K}_n \tag{3b}$$

$$\sum_{s\in\mathcal{S}_n} t_s \le T \tag{3c}$$

The objective is to minimize cell n's energy in content delivering such that in constraints (3b), the requested files in cell n are delivered, and in (3c) the total transmission duration is confined within a time limit T. Variable t_s is the time duration of using cluster s in content delivering. In constraints (3b), the associated user set $\mathcal{K}_n$ for SBS n and the set $\mathcal{F}_{nk}$, i.e., the user k's requested files to be served from SBS n, can be obtained from solving P1 or predetermined based on new inputs.

IV. Solution Development: Training Deep Neural Networks

In realistic networks, users' movement, channel conditions, and their file requests are varying frequently. Thus the developed optimization models P1 and P2 are expected to be solved efficiently such that the real-time decision making can be supported for network operations. Solving the combinatorial caching optimization problem P1 and the cluster scheduling problem P2 is in general with high computational complexity [2], [4], [5]. Although P2 is an LP, the problem's combinatorial aspect is exhibited in selection of users to form clusters [11]. To construct a cluster, any scheduler has to make binary decisions of determining whether or not each user should be included in a cluster. Moreover, the number of clusters increases exponentially with the user number K. For the scenario with large K, selecting optimal clusters among a large number of $2^K - 1$ clusters is computational heavy and time consuming. Thus directly applying conventional iterative optimization algorithms, e.g., Branch-and-Bound algorithm for P1, Simplex algorithm for P2, or other iterative heuristics for both problems, may have difficulties to meet the restricted delay and provide high-quality performance in real-time operations. To develop time-efficient algorithms for P1 and P2, especially for P2 due to its more stringent delay requirement in real-time online scheduling, we design a learning-based solution based on training DNNs.

We take P2 as an example in presenting the designed DNN. Ideally, we expect to train the DNN as a black-box algorithm that essentially works as a mapping from a problem instance to its corresponding optimal solution. However, doing so is impractical as the combinatorial nature of P2 poses high complexity in function approximation. This results in challenges to guarantee the optimality of the estimated solution. For instance, two solutions that only differ slightly in element values may lead to dramatically large gap on their corresponding objective values in P2. Though it is hard to directly use the DNN to precisely generate exact solutions for P2, it helps to identify the pattern of user channel conditions in relationship of optimal clustering strategies. The basic idea is to use DNN to predict the users that are most promising in terms of clustering in the optimal solution, according to the channel conditions and demands. In general, the DNN optimization procedure is divided into two phases, i.e., training and operation phases. During the training phase, the DNN is trained by data sets consisting of the parameters of P2 as the input, and the users participating in clustering in the optimal solution obtained by the solver as the desired output. With sufficient training, the DNN is able to accurately recognize the status of user clustering in the optimal solution. In this way, a considerably large amount of users are excluded, which significantly reduces the searching space of the optimization problem. Theoretically, as long as DNN can achieve precise

estimation, we can convert a large-scale instance of P2 to a much smaller one, without loss of optimality.

We design a DNN for P2 with an encoder for data regularization, and a hidden layer with a large number of nodes to further learn the input-output relations. One can see in Figure 1 for an illustrative structure. We adopt the resilient back-propagation (RProp) algorithm as the learning heuristic for supervised learning in the DNN training [6]. After training, given a new input to the well-trained DNN in the operation phase, an approximated solution can be efficiently obtained after simple operations via the DNN [6]. Specifically, the

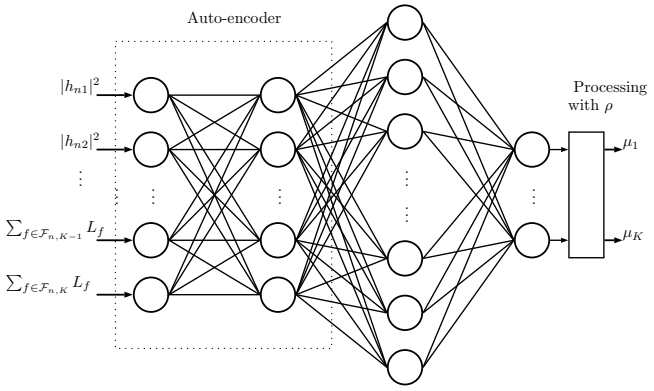

Figure 1. DNN structure.

DNN has an encoding layer, which consists of $2|\mathcal{K}_n|$ nodes. After encoding, 100 nodes are configured in the hidden layer. The number of inputs are $2|\mathcal{K}_n|$, consisting of $|\mathcal{K}_n|$ users' channel coefficients $|h_{nk}|^2,\ k \in \mathcal{K}_n$ and the users' requested data amount $\sum_{f\in\mathcal{F}_{nk}} L_f,\ k \in \mathcal{K}_n$ in cell n. The number of output nodes are $|\mathcal{K}_n|$. After processing, the network gives a $|\mathcal{K}_n|$-dimension binary vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_{K_n}]$ as the output. Each element μ_k indicates that whether or not user k participates in clustering scheduling. If user k is clustered, then μ_k is expected to be one, otherwise user k is scheduled by TDMA transmission with the output $\mu_k = 0$. The output of the DNN is a real-valued vector. We compute the mean value M for the vector's elements. Then, any element that is larger than ρM $(0 < \rho)$ is set to one in its corresponding position of $\boldsymbol{\mu}$ and is set to zero otherwise. The parameter ρ controls the number of positive elements in $\boldsymbol{\mu}$. The larger value the parameter ρ has, the fewer elements are set to be positive in $\boldsymbol{\mu}$. The training data is generated by solving the optimization problem for $|\mathcal{K}_n|$ users in each cell n. We extract the data samples in which there exists at least two users participating in clustering. In addition, all data samples used for training are chosen to be feasible for P2. We refer to the union of the sets used for DNN training, testing and validation as *training set* [6]. The data set is generated by solving the optimization problem P2 for $|K_n|$ users in each cell.

V. Performance Evaluation

We evaluate the performance of the developed DNN-based approach, in terms of computational time in DNN operation phase, performance gap between the DNN and the optimum, and training time in DNN training phase. In the network, 5 SBSs are deployed in the MBS's coverage area. For each SBS or the MBS, up to 20 users are connected. The network operates at 2 GHz. We use the COST-231-HATA path loss model, with the shadowing follows a Log-normal distribution with 3 dB standard deviation, and Rayleigh fading is considered. The parameter settings for the DNN structure are summarized in Table I. For illustration, we use a DNN based on $|\mathcal{K}_n| = 15$ in a cell to demonstrate the performance.

Table I
DNN SETTINGS

Parameter	**Value**
Input nodes	$2\|\mathcal{K}_n\|$
Encoder nodes	$2\|\mathcal{K}_n\|$
Hidden layer nodes	100
Output nodes	$\|\mathcal{K}_n\|$
Learning algorithm	RProp [6]

A. Comparison in Computational Time

We compare the CPU time (second) in computations between the DNN and the optimal algorithm. The results are shown in Table II. In terms of the optimization algorithm, we optimally solve P2 by applying the LP solver in Matlab, where the simplex algorithm or the interior-point algorithm is adopted. For a fair comparison, we evaluate the the DNN's running time per instance in Matlab, and also train the DNN in Matlab as well. The average computational time of the DNN per instance in its operation phase can be significantly reduced compared to directly applying the optimization algorithm. The CPU time of the optimal algorithm exponentially increases with the number of users, whereas the DNN is insensitive to the input size, and is able to deliver output efficiently.

Table II
CPU TIME IN COMPUTATION

Case	**DNN**	**Optimal Algorithm**
10 users	0.03	0.11
15 users	0.035	0.65
20 users	0.05	197

B. Performance of the DNN in approximating to the optimum

Figure 2 shows the performance of the DNN in approaching to the optimum in respect of the size of the training set. We use a metric called "optimality rate" to measure to what extend we can keep the optimality of the solution obtained with DNN. For example, the value of "0.8" in the y-axis means that the solutions obtained by the DNN are with 80% probability to be global optimal. For performance evaluation in Figure 2, two sets are respectively used for computing the optimality rate, i.e., the set that we used for training the DNN, and the whole data set (with 42000 data points in total). The size of the training set ranges from 100 to 10000. In Figure 2, the optimal size for training is around 2000, leading to the highest optimality rate around 98% As expected, when the

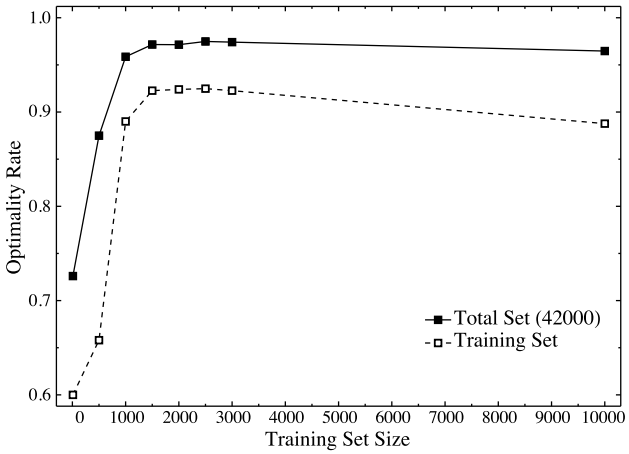

Figure 2. Optimality in function of training set size.

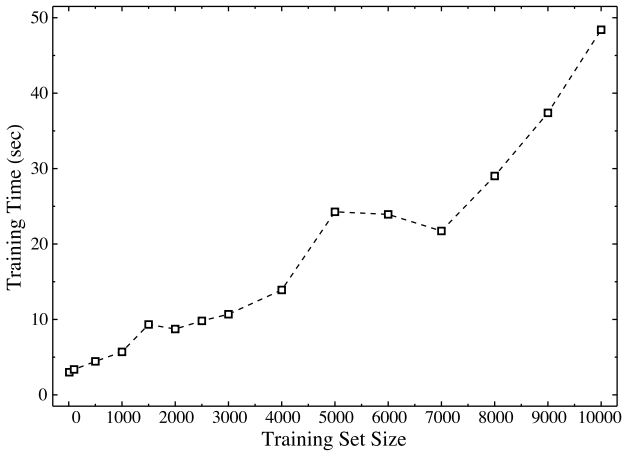

Figure 3. Training time in function of the training set size.

size of the training set is small, one cannot expect a high-quality approximation performance. In other words, the DNN is unable to learn the optimization process without being trained with sufficient data samples. In Figure 2, we observe that with a small training set (i.e. hundreds of samples), the DNN can achieve an optimality rate above 70% on the total set. Furthermore, in Figure 2 if we compare the objective value between the DNN and the optimal algorithm, in average the DNN can achieve 90% approximation to the optimum.

C. Training Time

This subsection evaluates the required the DNN training time with respect to the training set size. As shown by Figure 3, computationally, the training time is linearly scalable with the training set size. On the other hand, the training time may not monotonically increase with the size of the training set. The training set includes the set used for validation during the training process. Thus, by enlarging the set, the validation set is enlarged as well. This may cause the training process to terminate than before. Recall that in Figure 2 the training set with size 2000 leads to the highest optimality rate. One can observe in Figure 3 that the training process is completed around 10 seconds with the set size 2000. Even with 10000 samples in the set, the training can be finished within 1 minute.

VI. Conclusions

We considered energy optimization in a cache-enabled heterogeneous network. We formulated optimization problems of caching placement, user association, and content delivering in order to improve the network performance. Aiming at addressing the issue of computational efficiency in real-time network operations, we considered a learning-based approach, and designed a DNN to train the iterative algorithms, such that time-efficient and near-optimal solutions can be obtained. Numerical results demonstrate that the well-trained DNN can progressively achieve around 90% approximation to the optimum, and with promising performance in supporting online optimization.

VII. Acknowledgments

The work has been supported by the European Research Council (ERC) project "Actively Enhanced Cognition based Framework for Design of Complex Systems (AGNOSTIC)"

References

[1] G. Paschos, E. Bastug, I. Land, G. Caire and M. Debbah, "Wireless caching: technical misconceptions and business barriers," in IEEE Communications Magazine, vol. 54, no. 8, pp. 16-22, Aug. 2016.

[2] W. Jiang, G. Feng and S. Qin, "Optimal Cooperative Content Caching and Delivery Policy for Heterogeneous Cellular Networks," in IEEE Transactions on Mobile Computing, vol. 16, no. 5, pp. 1382-1393, May 2017.

[3] A. Khreishah, J. Chakareski and A. Gharaibeh, "Joint Caching, Routing, and Channel Assignment for Collaborative Small-Cell Cellular Networks," in IEEE Journal on Selected Areas in Communications, vol. 34, no. 8, pp. 2275-2284, Aug. 2016.

[4] K. Poularakis and L. Tassiulas, "On the Complexity of Optimal Content Placement in Hierarchical Caching Networks," in IEEE Transactions on Communications, vol. 64, no. 5, pp. 2092-2103, May 2016.

[5] L. Lei, D. Yuan, C. K. Ho, and S. Sun, "A unified graph labeling algorithm for consecutive-block channel allocation in SC-FDMA, IEEE Trans. Wireless Commun., vol. 12, no. 11, pp. 5767-5779, Nov. 2013

[6] V. Sze, Y. Chen, T. Yang, and J. Emer, "Efficient processing of deep neural networks: A tutorial and survey", http://arxiv.org/abs/1703.09039, Mar. 2017.

[7] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. "Resource Management with Deep Reinforcement Learning", In Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets). ACM, New York, USA, 50-56, 2016.

[8] W. Wang, R. Lan, J. Gu, A. Huang, H. Shan, Z. Zhang, "Edge Caching at Base Stations with Device-to-Device Offloading", in IEEE Access, vol.PP, no.99, pp.1-1

[9] L. Lei, D. Yuan, C. K. Ho, and S. Sun, "Optimal cell clustering and activation for energy saving in load-coupled wireless networks," in *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6150–6163, Nov. 2015.

[10] K. Murty, *Linear programming*, Wiley, 1983.

[11] V. Angelakis, A. Ephremides, Q. He and D. Yuan, "Minimum-Time Link Scheduling for Emptying Wireless Systems: Solution Characterization and Algorithmic Framework," in *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1083-1100, Feb. 2014.