# Distributed Caching in $5G$ Networks: An Alternating Direction Method of Multipliers Approach

Azary Abboud[◇,†], Ejder Başţuğ[◇], Kenza Hamidouche[◇], and Mérouane Debbah[◇,⋆]

[◇]Large Networks and Systems Group (LANEAS), Supélec, 91192, Gif-sur-Yvette, France
[†]Automatic and Control Department, Supélec, 91192, Gif-sur-Yvette, France
[⋆]Mathematical and Algorithmic Sciences Lab, Huawei France R&D, Paris, France
{azary.abboud, ejder.bastug, kenza.hamidouche, merouane.debbah}@supelec.fr

***Abstract*—We consider the problem of distributed caching in next generation mobile cellular networks (a.k.a., $5$G) where densely-deployed small base stations (SBSs) are able to store and deliver users' content accordingly. In particular, we formulate the optimal cache allocation policy as a convex optimization problem where a subset of SBSs have their own $i$) local cost function which captures backhaul consumption aspects in terms of bandwidth and $ii$) a set of local network parameters and storage constraints. Given the fact that no coordination involves between SBSs, we then solve this problem distributively using the Alternating Direction Method of Multipliers (ADMM) approach. The proposed ADMM-based algorithm relies on the application of random Gauss-Seidel iterations on the Douglas-Rachford splitting operator, which results in a low-complexity and easy-to-implement solution for SBSs. We examine the convergence of our proposed algorithm via numerical simulations with different parameters of interest such as storage capacity distribution of SBSs, content catalogue size, demand intensity and demand shape. Our numerical results show that the proposed algorithm performs well in terms of convergence and requires less iterations as the number of contents in the catalogue increases.**

***Index Terms*—$5G$, distributed caching, optimal cache allocation, ADMM, distributed convex optimization, wireless networks**

## I. Introduction

Ever growing demand of mobile users [1] is reshaping discussions both in industry and academia, pushing the current mobile infrastructure to evolve towards next generation (a.k.a. $5G$) mobile cellular networks [2]. One of the candidate solutions to satisfy this demand and offload the backhaul is to proactively store users' contents at the edge of the mobile network, either in base stations or user terminals [3].

Indeed, although the idea of putting users' contents in cache-enabled nodes of the cellular networks is somewhat recent, many works have addressed the caching problem from different aspects resulting in an extensive literature. For instance, predicting users' behaviour and proactively storing their contents at small base stations (SBSs) is studied in [4], whereas the benefit of proactive caching in a mobility setup is exploited in [5]. A coded caching scheme using information theoretic arguments is given in [6], whereas a similar scheme but with multi-level architecture is studied in [7]. Performance evaluation of coded caching gains setups can be found in [8]–[10]. From a game theoretic standpoint, a many-to-many matching game formulation which takes into consideration the content dissemination in social networks is shown in [11]. Under a given estimation of backhaul usage via collaborative filtering (CF), a one-to-many matching game between SBSs and user terminals (UTs) is formulated in [12]. Additionally, a learning based online caching scheme is presented in [13]. The caching gains in stochastically distributed cache-enabled SBSs are studied in [14], [15]. These studies point out the importance of caching in $5G$ wireless networks, and provide their investigations both from performance and algorithmic perspectives. However, the practical efforts for these dense networks are still in its infancy mainly due to lack of easy-to-implement distributed solutions. We refer readers to [16] for a recent survey.

Given the motivations above, our main contribution in this work is to formulate the cache allocation policy as a convex optimization problem and provide a distributed algorithm implemented at each cache-enabled SBS. More specifically, we define a global convex cost function as the sum of local convex functions of users' demand and network topology (i.e., physical connection between SBSs and UTs) and the linear constrains are given on storage size due to the resource-limited SBSs. To solve this problem, we adopt a similar approach to [17] which uses the ADMM in the context of optimal flow in smart grids. We then provide a distributed caching algorithm which relies on the application of random Gauss-Seidel iterations on the Douglas-Rachford splitting operator [18]. This allows each SBS to solve its given sub-problem via this low-complexity iterative algorithm by taking into account users' demand, network topology and storage constraint.

Briefly, the motivation of using such a distributed approach is to 1) avoid the communication overhead between SBSs and the central scheduler (CS) that is in charge of the decision mechanism, and 2) distribute the computational burden of the CS among SBSs. Indeed, the origin of distributed optimization

This research has been supported by the ERC Starting Grant 305123 MORE (Advanced Mathematical Tools for Complex Network Engineering), the projects 4GinVitro and BESTCOM.

techniques dates back to the seminal work of Tsitsiklis and Bertsekas [19]. Among extensive studies on these techniques which are not covered in this work due to the lack of space, ADMM is shown to promise faster convergence at some negligible cost of synchronization and coordination compared to its alternatives [18].

The rest of the paper is organized as follows. Section II details the network model under consideration. The optimal cache allocation policy is formulated globally in Section III. This formulation is then adapted to the local case in Section IV for solving the problem distributively at the SBSs and the ADMM is introduced for that purpose. Numerical results which validate the convergence of the ADMM-based algorithm are presented in Section V and the impact of various system parameters on the convergence are discussed. We finally draw our conclusions in Section VI and give possible future directions accordingly.

The notation used in the rest of the paper is given as follows. Lower and upper case italic symbols (e.g., $a$, $A$) represent scalar values. Lower case boldface symbols (e.g., $\mathbf{b}$) denote row vectors whereas upper case boldface symbols (e.g., $\mathbf{A}$) are matrices. The indicator function $\mathbb{1}\{\cdot\}$ returns 1 when the statement in the argument holds, and 0 otherwise. The transpose of $\mathbf{a}$ is denoted by $\mathbf{a}^T$.

## II. Network Model

Let us consider a network consisting of a set of $M$ SBSs and $N$ users denoted by $\mathcal{M} = \{1, ..., M\}$ and $\mathcal{N} = \{1, ..., N\}$ respectively. In this setup, the SBSs are connected to a CS via limited backhaul links with the purpose of providing broadband Internet connection to their users. The wireless downlink rates from the SBSs to users are given by the matrix $\mathbf{R}$ of dimension $M \times N$, where each entry $R_{m,n}$ denotes the achievable rate from SBS $m$ to user $n$. For simplicity, we assume that communication in the downlink is done in a time-division duplexing (TDD) manner with multiple frequency blocks (i.e., orthogonal frequency-division multiple access (OFDMA)), thus intra-cell and inter-cell interference are avoided. Then, the *wireless connectivity matrix* $\mathbf{C} \in \{0,1\}^{M \times N}$, representing the connections between the SBSs and users is structured as

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_M \end{bmatrix} = \begin{bmatrix} c_{1,1} & \dots & c_{1,N} \\ \vdots & \ddots & \vdots \\ c_{M,1} & \dots & c_{M,N} \end{bmatrix}, \tag{1}$$

where $c_{m,n} = \mathbb{1}\{R_{m,n} \geq R'\}$. The purpose of the target bitrate $R'$ constraint for connectivity is to guarantee a certain quality-of-service (QoS) in the downlink. Each user $n$ with a wireless link rate $R_{m,n}$ below this threshold is assumed to be not connected to the SBS $m$.

In our model, we assume that the SBSs have storage units with capacities $\mathbf{s} = [s_1, ..., s_M] \in \{\mathbb{Z}^+\}^{1 \times M}$. These storage capacities in the decreasing ordered case follow a Zipf-like distribution $P_{\mathbf{s}}(s, \alpha)$ defined as [20]

$$P_{\mathbf{s}}(s, \alpha) = \frac{\Omega}{s^\alpha} \tag{2}$$

with

$$\Omega = \Big( \sum_{i=1}^{M} \frac{1}{i^\alpha} \Big)^{-1},$$

where the parameter $\alpha$ characterizes the steepness of the distribution. In fact, the evidence of such a law is shown for the content distribution in web proxies [20], whereas in our case we also treat for modelling the storage capacity distribution. By having storage capabilities at the SBSs, contents can be cached in order to serve users' predicted requests locally, and thus reduce backhaul usage and access delays. The sketch of the considered network model is given in Fig. 1.

In the following, we suppose that the users' content demand arrival, over $T$ time slots, is modeled by a Poisson process with rate/intensity parameter $\lambda$. Additionally, we suppose that users' demands are made from a *catalogue* of $F$ distinct contents. The length of each file is denoted by $Q < s_m, m = 1, ..., M$. We assume that the content popularity distribution of users' drawn from the catalogue is characterized by another Zipf law with parameter $\beta$. Given the arrival process and content popularity distribution, the users' content demand counts are represented by the *user demand matrix* $\mathbf{D}^u \in \{\mathbb{N}\}^{N \times F}$. Then, the whole content demands observed at the SBSs level is given by

$$\mathbf{D} = \mathbf{C}\mathbf{D}^u \in \{0,1\}^{M \times F}. \tag{3}$$

For ease of exposition, we assume that the matrix $\mathbf{D}$ is perfectly known. Note that, in practice, the demand of users (and its observation at SBSs) are correlated and can be predicted up to a certain level, i.e., using statistical inference tools from machine learning [3].

Suppose that the *cache indicator matrix* of SBSs, denoted as $\mathbf{X} \in \{0,1\}^{M \times F}$, is given as follows

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_M \end{bmatrix} = \begin{bmatrix} x_{1,1} & \dots & x_{1,F} \\ \vdots & \ddots & \vdots \\ x_{M,1} & \dots & x_{M,F} \end{bmatrix}, \tag{4}$$

where the entry $x_{m,f}$ is 1 if SBS $m$ caches content $f$, and 0 otherwise. As alluded earlier, the aim is to decide which contents have to be cached in the SBSs while minimizing the cost function representing the cost of serving users using the backhaul link. In other words, the cache indicator matrix has to be optimized for a given cost and storage constraints of SBSs. However, to achieve this goal, the cost in the sense of backhaul usage has to be defined properly by taking into account users' content demand statistics and network topology. In what follows, we define this cost function and formulate the problem accordingly.

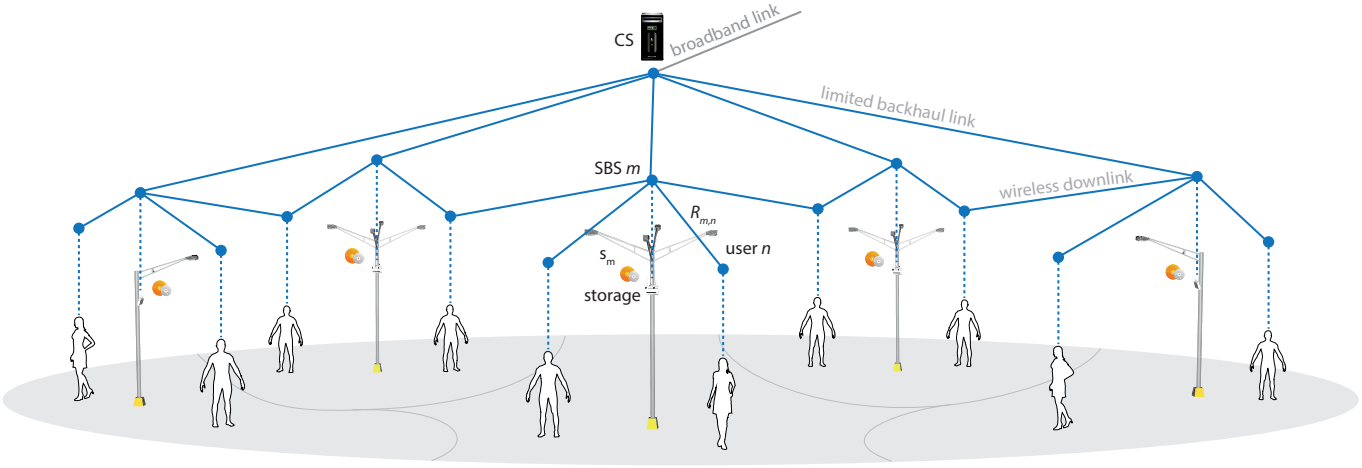


Figure 1: An illustration of the scenario which consists of $M$ cache-enabled SBSs and $N$ users.

## III. Optimal Cache Allocation Policy

In this section, we start by defining the cost function which represents the price of serving all users in addition to the price of taking wrong decisions at the SBSs and caching content that would not be requested by users. Depending on which content is cached and which one is expected to be requested, four different cases can be incorporated into the cost function as follows:

1) *Part of the content in the catalogue is* **cached** *and is going to be* **demanded***:* The part of the content in this case is fetched in advance via the backhaul once. Therefore, the induced cost is given by

$$\eta_m \left[ \mathbf{x}_m \mathbf{g}_m^T \right], \tag{5}$$

where $\eta_m$ is an arbitrary scalar coefficient that represents the price of one unit backhaul usage and $\mathbf{x}_m$ is the cache indicator (allocation) vector for SBS $m$. Moreover, $\mathbf{g}_m \in \{0,1\}^{1\times F}$ is the demand indicator vector for the same SBS, where the entries are computed as follows

$$g_{m,f} = \begin{cases} 1 & \text{if } d_{m,f} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

2) *Part of the content in the catalogue is* **cached** *but is* **not** *going to be* **demanded**: The backhaul is used unnecessarily, hence, cost is computed as follows:

$$\eta_m \left[ \mathbf{x}_m \bar{\mathbf{g}}_m^T \right], \tag{7}$$

where $\bar{\mathbf{g}}_m$ is the complementary of $\mathbf{g}_m$ with the entries given such as

$$\bar{g}_{m,f} = \begin{cases} 0 & \text{if } g_{m,f} = 1 \\ 1 & \text{otherwise.} \end{cases} \tag{8}$$

3) *Part of the content is* **not cached** *but is going to be* **demanded**: The content has to be fetched via the backhaul on each request, yielding the following cost

$$\eta_m \left[ \bar{\mathbf{x}}_m \mathbf{d}_m^T \right], \tag{9}$$

where $\bar{\mathbf{x}}$ is the complementary of $\mathbf{x}_m$, defined in a similar way as $\mathbf{g}_m$.

4) *Part of the content in the catalog is* **not cached** *and is* **not** *going to be* **demanded**: No operation cost is involved in this case.

Thus, combining all the cases above, the optimization problem for finding the optimal cache allocation vector $\mathbf{x}_m$ under given storage constraint $s_m$ is formulated as

$$\begin{aligned} \underset{\mathbf{x}_m, m=1\dots M}{\text{minimize}} \quad & f(\mathbf{x}) = \sum_m \eta_m \left[ \mathbf{x}_m \mathbf{g}_m^T + \mathbf{x}_m \bar{\mathbf{g}}_m^T + \bar{\mathbf{x}}_m \mathbf{d}_m^T \right] \\ \text{subject to} \quad & ||\mathbf{x}_m||_1 \leq s_m, m = 1 \dots M. \end{aligned} \tag{10}$$

In this optimization problem, note that $\bar{\mathbf{x}}_m$ and $\bar{\mathbf{g}}_m$ can be written as $\bar{\mathbf{x}}_m = \mathbf{1}_F - \mathbf{x}_m$ and $\bar{\mathbf{g}}_m = \mathbf{1}_F - \mathbf{g}_m$ respectively, where $\mathbf{1}_F$ is the vector with all elements equal to one. Therefore, the global cost function is reduced to

$$\begin{aligned} f(\mathbf{x}) &= \sum_m f_m(\mathbf{x}_m) \\ &= \sum_m \eta_m (\mathbf{x}_m \mathbf{1}_F^T - \mathbf{x}_m \mathbf{d}_m^T) + \eta_m \mathbf{1}_F \mathbf{d}_m^T, \end{aligned}$$

where $f_m(\mathbf{x}_m)$ is the cost function of the SBS $m$. In fact, the function $f(\mathbf{x})$ is separable in $m$ and can be straightforwardly proved as in [21]. Additionally, observe that the entries of $\mathbf{x}$ take binary values $0-1$, thus might fall the optimization problem into a mixed-integer program which is non-tractable in practice. However, we avoid this situation implicitly by having sufficiently small and equally-sized contents (or one can alternatively introduce coding as in [8]).

Conventionally, the problem (10) can be solved at the CS in a centralized way by

i) collecting users' demands from the SBSs as well as the storage capacities of SBSs;

ii) solving the problem accordingly;
iii) transferring the adequate cache indicator vector to each SBS.

This centralized optimization scheme however induces latency in the computation and communication burden. Alternatively, the solution of (10) can be obtained efficiently by using distributed convex optimization solvers. In the next section, we detail the steps of the ADMM-based approach, which in turn will allow us to handle the problem at each SBS distributively.

## IV. Alternating Direction Method of Multipliers

In a nutshell, ADMM [18] is a decomposition-coordination procedure that was initially introduced by Gabay, Mercier et al. in the mid-1970. In our context, the ADMM is used to solve the optimal cache allocation problem distributively after dividing it into several sub-problems, each solved by a processor embedded in a corresponding SBS. In order to achieve this goal, the problem has to be reformulated into the following form [17]

$$\begin{aligned} &\text{minimize} \quad h(\boldsymbol{y}) + g(\boldsymbol{z}) \\ &\text{subject to} \quad \boldsymbol{B}\boldsymbol{y} = \boldsymbol{z}, \end{aligned} \tag{11}$$

where $h$ and $g$ are two convex functions that we further identify in the next subsection, $\boldsymbol{y} \in \mathbb{R}^L$ is the primal variable we seek to find, and $\boldsymbol{z} \in \mathbb{R}^{LM}$ are the slack variables that represent along with $\boldsymbol{\lambda} \in \mathbb{R}^{LM}$ the state of ADMM. $\boldsymbol{\lambda}$ is the set of Lagrangian multipliers associated with the constraints $\boldsymbol{B}\boldsymbol{y} = \boldsymbol{z}$. We form the augmented Lagrangian function as

$$\begin{aligned} \mathcal{L}_\rho(\boldsymbol{y}, \boldsymbol{z}; \boldsymbol{\lambda}) \triangleq h(\boldsymbol{y}) + g(\boldsymbol{z}) + \langle \boldsymbol{\lambda}, \boldsymbol{B}\boldsymbol{y} - \boldsymbol{z} \rangle \\ + \frac{\rho}{2} \left\| \boldsymbol{B}\boldsymbol{y} - \boldsymbol{z} \right\|^2, \end{aligned} \tag{12}$$

where $\rho > 0$ is a penalty parameter and $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^T \boldsymbol{v}$ is the inner product of $\boldsymbol{u}$ and $\boldsymbol{v}$. Then, the ADMM method for solving (11) consists of $i$) an $\boldsymbol{y}-$minimization step, $ii$) a $\boldsymbol{z}-$minimization step and $iii$) a dual variable $\boldsymbol{\lambda}$ update step, given as

$$\boldsymbol{y}^{k+1} = \underset{\boldsymbol{y}}{\operatorname{argmin}} \ \mathcal{L}_\rho(\boldsymbol{y}, \boldsymbol{z}^k; \boldsymbol{\lambda}^k), \tag{13a}$$

$$\boldsymbol{z}^{k+1} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \ \mathcal{L}_\rho(\boldsymbol{y}^{k+1}, \boldsymbol{z}; \boldsymbol{\lambda}^k), \tag{13b}$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\boldsymbol{B}\boldsymbol{y}^{k+1} - \boldsymbol{z}^{k+1}). \tag{13c}$$

### A. Distributed Caching with ADMM

As mentioned previously, a form similar to (11) is required for solving the caching problem distributively. This is first done by introducing slack variables $sl_m, m = 1 \ldots, M$ to transform the inequality constraints in (10) into equality constraints. When we append these slack variables to the primal variable vector $\mathbf{x}$, we obtain a new vector $\boldsymbol{y}$ of size $L = M(F+1)$. Each sub-block $\boldsymbol{y}_m$ of $\boldsymbol{y}$ corresponds to SBS $m$ and consists of the vector $\boldsymbol{x}_m$ as well as the slack variable $sl_m$. At this point, the optimization problem can be reformulated as

$$\begin{aligned} &\text{minimize} \quad h(\boldsymbol{y}) \\ &\text{subject to} \quad \boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}, \end{aligned} \tag{14}$$

where

$$\begin{aligned} \boldsymbol{y} =& [\boldsymbol{x}_1; sl_1; \ldots; \boldsymbol{x}_M; sl_m] \\ =& [x_{11}, x_{12}, \ldots, x_{1F}, sl_1, \ldots, x_{M1}, \ldots, x_{MF}, sl_M]^T, \end{aligned}$$

$$h(\boldsymbol{y}) = \sum_m h_m(\boldsymbol{y}_m),$$

$$h_m(\boldsymbol{y}_m) = f_m(\boldsymbol{x}_m),$$

$$\boldsymbol{b} = [s_1; \ldots; s_M]^T \in \mathbb{R}^M,$$

and the matrix $\boldsymbol{A} \in \mathbb{R}^{M \times L}$ which regroups the constraints is given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & \ldots & 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & \ldots & 0 & 0 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}.$$

Then, we finally obtain the similar form as in (11) by inserting the vector of slack variables $\boldsymbol{z} \in \mathbb{R}^{LM \times 1}$ and using the indicator function

$$g(\boldsymbol{z}) = \begin{cases} 0 & \text{if } \sum_j z_{mj} = b_m, \quad \forall m = 1, \ldots M \\ +\infty & \text{otherwise.} \end{cases}$$

The matrix $\boldsymbol{B}$ that regroups the reformulated constraints is given by

$$\boldsymbol{B} = \begin{bmatrix} \text{diag}(\boldsymbol{A}_1) \\ \vdots \\ \text{diag}(\boldsymbol{A}_M) \end{bmatrix},$$

where

$$\text{diag}(\boldsymbol{A}_1) = \begin{bmatrix} a_{11} & 0 & \ldots & 0 \\ 0 & a_{12} & \ldots & 0 \\ \ldots & \ldots & \ddots & \ldots \\ 0 & \ldots & 0 & a_{1L} \end{bmatrix}.$$

Let $\boldsymbol{\lambda} \in \mathbb{R}^{LM}$ be the vector of Lagrangian multipliers associated with the set of constraints $\boldsymbol{B}\boldsymbol{y} = \boldsymbol{z}$. Following the result of [17], one an show that the ADMM equations reduce to

$$x_{mj}^{k+1} = \left[ x_{mj}^k - \frac{1}{a_{mj}^2} \left[ \frac{1}{\rho} (\eta_m (1 - d_{mj}) + a_{mj} \pi_m^k) + \frac{a_{mj} r_m(\boldsymbol{x}^k)}{d(m)} \right] \right]_+, \tag{15}$$

$$z_{mj}^{k+1} = a_{mj} x_{mj}^{k+1} - \frac{1}{d(m)} r_m(\boldsymbol{x}^{k+1}), \tag{16}$$

$$\pi_m^{k+1} = \pi_m^k + \frac{\rho}{d(m)} r_m(\boldsymbol{x}^{k+1}), \tag{17}$$

where $[x]_+ = \max\{0, x\}$, $r_m(\boldsymbol{x}^k) = \sum_j a_{mj} x_{mj}^k$ is the residual of the $m^{th}$ constraint, and $d(m)$ represents the number of non-zero elements in the $m^{th}$ constraint. $\pi_m$ is the Lagrangian multiplier associated with $\sum_j z_{mj} = b_m$. In [17], it was proved that $\lambda_{mj} = \pi_m, \forall j = 1 \dots L$ which reduces the complexity of the algorithm by decreasing the number of Lagrangian multipliers from $L$ multipliers to only one multiplier by SBS.

In practice, these update steps are straightforward to implement and do not introduce high computational complexity at the SBSs. Note that the $z_{mj}$ step can be omitted as it is not involved either in the update of $x_{mj}$ or $\pi_m$. Therefore, the ADMM method applied to the optimal cache allocation policy reduces into two update steps. The first update step (15) is on the primal variable $\boldsymbol{x}_m$ it represents which contents have to be cached in SBS $m$. The second step (17) updates the Lagrangian multiplier $\pi_m$ of the same SBS. In other words, given the storage constraint $s_m$ and demand vector $\mathbf{d}_m$, each SBS $m$ has to solve its own sub-problem using equations (15) and (17). We would like to note that no synchronisation or coordination are required between the SBSs, as there are no parameters shared between them. The procedure is summarized in Algorithm 1. The complexity of the algorithm is mainly driven by the number of iterations and number of contents in the catalogue, thus $\mathcal{O}(\kappa F)$ where $\kappa$ the sufficient number of iterations to satisfy the convergence to optimal values with some precision. Indeed, the values of $\kappa$ depends on various parameters such as storage capacity distribution, catalog size, demand intensity and demand shape. These parameters are numerically examined in the following.

**Algorithm 1: Distributed Caching using ADMM**

1) Initialize $x_{mj}$ and $\pi_m$ to the initial values $x_{mj}^0$ and $\pi_m^0$.
2) At iteration $k+1$:
   a) Every SBS $m = 1, ..., M$ decides whether to cache a content or not by computing for every component $x_{mj}$

$$x_{mj}^{k+1} = \left[ x_{mj}^k - \frac{1}{a_{mj}^2} \left[ \frac{1}{\rho} (\eta_m (1 - d_{mj}) + a_{mj} \pi_m^k) + \frac{a_{mj} r_m(x^k)}{d(m)} \right] \right]_+$$

   b) Every SBS $m = 1, ..., M$ updates the Lagrangian multiplier of its constraint

$$\pi_m^{k+1} = \pi_m^k + \frac{\rho}{d(m)} r_m(\boldsymbol{x}^{k+1})$$

3) For each SBS apart, if its constraint is still violated, increase $k$ and go to step 2. Otherwise, SBS $m$ stores the files as given by $x_{mj}, j = 1, \dots, F$.

## V. Numerical Results

In this section, we evaluate the performance of the proposed caching mechanism in terms of convergence and study the impact of different parameters on the global cost. More specifically, we study the variation of the global cost with respect to the storage capacity distribution of SBSs, the demand shape, the demand intensity and the number of contents in the catalogue. The simulation setup is repeated several times and the obtained results are averaged out. For ease of exposition, we assume that each user is connected to only one SBS and user connections to SBSs are done uniformly at random. The list of simulation parameters is given in Table I.

Table I: Simulation Parameters.

| Parameter | Description | Default Value |
|---|---|---|
| $\alpha$ | Storage distribution parameter | 0.1 |
| $F$ | Catalog size | 128 contents |
| $\lambda$ | Demand intensity | 64 requests/sec |
| $\beta$ | Demand shape parameter | 0.1 |
| $M$ | Number of SBSs | 8 |
| $N$ | Number of users | 128 |
| $T$ | Max. time for demand generation | 128 seconds |
| $s_m$ | Storage of SBS $m$ | $0 \sim F$ MBit |
| $\eta_m$ | Cost coefficient of SBS $m$ | 2 |

In all the figures related to global cost, we additionally show the global cost of *optimal* offline solution for comparison purposes. Indeed, the optimality is in the sense of storing most popular content greedily under given demand information and storage constraints. This offline approach has $\mathcal{O}(F \log F)$ worst-case complexity for each SBS, mainly due to the sorting operation of the contents [4].

We would like to note that, as shown in all the figures, whatever is the parameters studied in numerical setup, the ADMM-based algorithm converges as the number of iterations increases. For exposition purposes, we restrict ourselves to plot the evolution of the mean global cost and constraints violations for the first 200 iterations. Indeed, depending on the parameters setting, the number of iterations for convergence changes. Moreover, we observe that the mean storage constraints are never violated at the SBSs. In other words, the number of cached content at the SBSs does not exceed their storage capacity. This is shown in the violation plots, where the difference between the storage capacity and the cached content converges to zero. In the following, we discuss the impact of parameters of interest in details.

*1) Impact of storage capacity distribution ($\alpha$):* First, to show the impact of the storage size at the SBSs on the global cost, we assume that the storage capacities of SBSs are random and follow a Zipf distribution with parameter $\alpha$ as mentioned before. Three different values of the shape parameter $\alpha \in \{0.1, 0.4, 0.8\}$ are considered in order to study

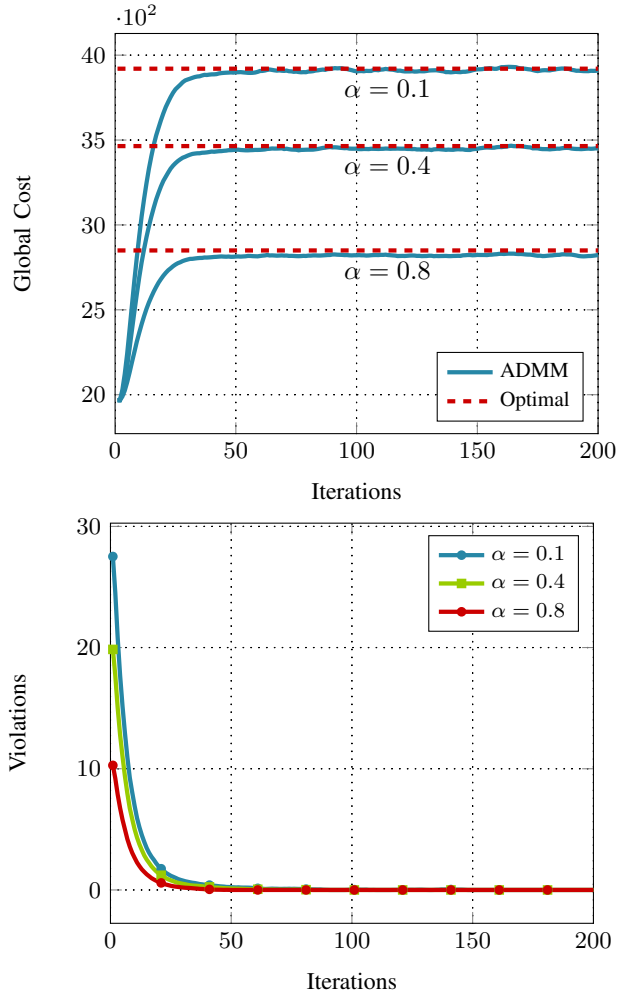


Figure 2: Impact of demand storage capacity distribution on the global cost and number of constraint violations.

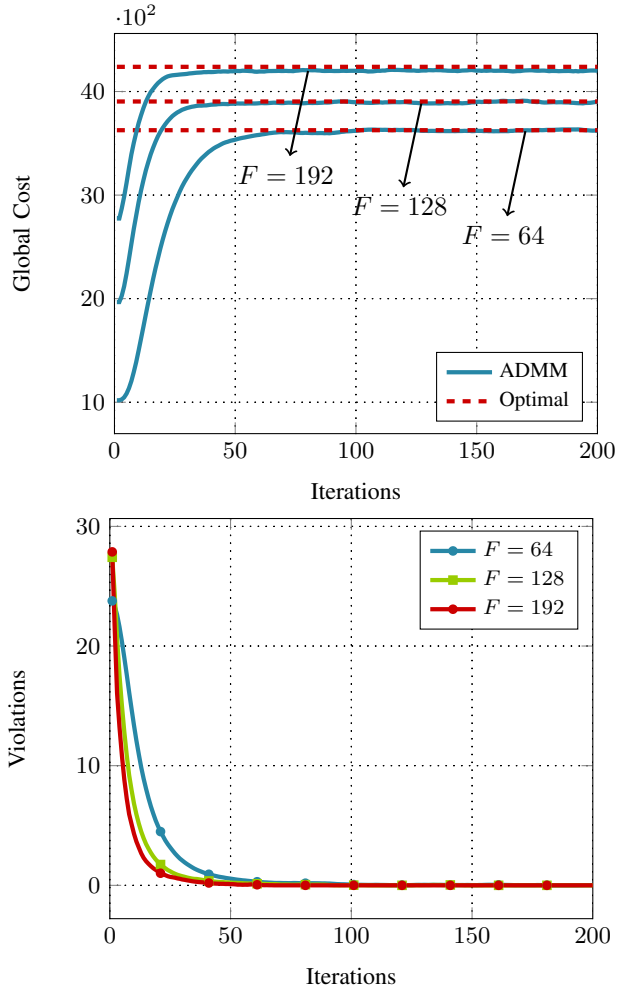


Figure 3: Impact of content catalog size on the global cost and number of constraint violations.

its effect on the cost. It should be noted here that the storage capacities of the SBSs are more uniform and heterogeneous in small values of $\alpha$, whereas high values of $\alpha$ correspond to more homogeneous storage units with high capacities. Fig. 2 shows the changes of the global cost with respect to the number of iterations for different values of $\alpha$. From the figure, we can see that when the storage capacity and homogeneity of the SBSs increase (i.e., when $\alpha$ increases), the global cost decreases. In other words, the cost is reduced with increment of $\alpha$, as less contents are served from the core network through the backhaul links.

*2) Impact of catalog size ($F$):* Fig. 3 shows the impact of the number of distinct contents on the global cost. We consider three values for the number of contents $F \in \{64, 128, 192\}$. The variation of the global cost for these values shows that the load on the backhaul links increases as the number of considered contents increases. This is somewhat obvious since a higher number of distinct contents induces a higher number of distinct demands. Therefore, due to the limited storage capacity of SBSs, most of the contents are served via backhaul links. One interesting observation here is that, as the number of contents increases, the ADMM-based algorithm converges faster (i.e., $\kappa$ decreases), yielding less iterations thus outperforming the optimal offline approach. This is indeed useful in practice as the SBSs are deployed in densely populated urban-areas inducing requests for a large number of contents.

*3) Impact of demand intensity ($\lambda$):* Similarly Fig. 4 plots the global cost by varying the intensity of the demand in the network. We consider three different values of demand intensity $\lambda \in \{100, 200, 300\}$, where $\lambda$ represents the average number of requests per second. The figure shows that as the number of requests becomes higher, the global cost increases as well. In other words, high demand intensity in the network introduces high traffic on the backhaul links due to the fact that these requests for non-cached contents are satisfied via backhaul links rather than the caches of SBSs.

*4) Impact of demand shape ($\beta$):* Finally, in order to study the impact of demand shape, we consider that user requests follow a Zipf distribution with parameter $\beta \in \{0.1, 0.2, 0.4\}$. Fig. 5 shows that the global cost is higher for small values of

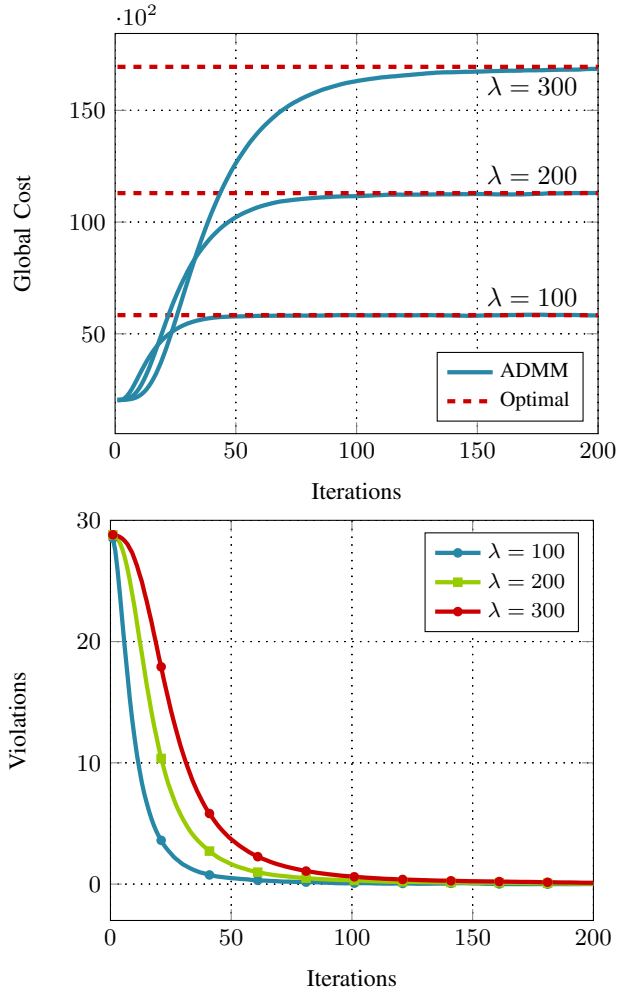


Figure 4: Impact of demand intensity on the global cost and number of constraint violations.

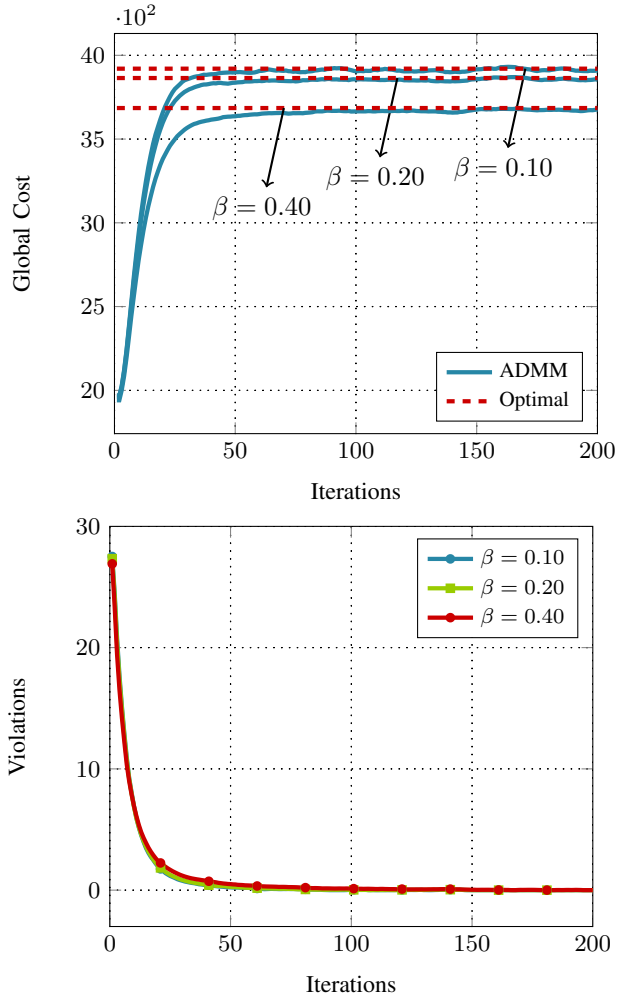


Figure 5: Impact of demand shape on the global cost and number of constraint violations.

$\beta$ and decreases by increasing the value of $\beta$. In fact, when the shape parameter $\beta$ is high, most of the requests are generated for a small subset of contents, thus, most of users' requests can be served locally from the SBSs when those contents are cached, without using backhaul links, which results in a low global cost.

## VI. Conclusions

We proposed a novel ADMM-based distributed caching approach for cache-enabled SBSs. We examined the convergence of our approach via numerical simulations under various parameter settings and showed that the amount of required iterations decreases as the catalogue size increases. With this in mind, our approach can be used as an iterative and distributive solution for solving cache allocation problem in SBSs.

An interesting future work would be to conduct a more rigorous analysis on its *convergence* rather than using numerical simulations. Yet another line of work would be considering a *complex topology* in which users are connected to multiple base stations (i.e., SBSs and macro cell). In this case, our ADMM-based approach can be studied under *limited-coordination* between the SBSs due to the need of parameter sharing for common users. Additionally, *cache-enabled device-to-device (D2D) communications* can be taken into consideration. This would require a different technical treatment as D2D communications have *asynchronous* behaviour.

## References


[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019," *White Paper, [Online] http://goo.gl/k84Qpo*, 2015.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5G be?" *arXiv preprint: 1405.2957*, 2014.

[3] E. Baştuğ, M. Bennis, and M. Debbah, "Living on the Edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, August 2014.

[4] E. Baştuğ, J.-L. Guénégo, and M. Debbah, "Proactive small cell networks," in *ICT 2013*, Casablanca, Morocco, May 2013.

[5] V. A. Siris, X. Vasilakos, and G. C. Polyzos, "Efficient proactive caching for supporting seamless mobility," *arXiv preprint: 1404.4754*, 2014.

[6] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[7] J. Hachem, N. Karamchandani, and S. Diggavi, "Coded caching for heterogeneous wireless networks with multi-level access," *arXiv preprint: 1404.6560*, 2014.

[8] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE INFOCOM*. IEEE, 2012, pp. 1107–1115.

[9] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *arXiv preprint: 1305.5216*, 2013.

[10] J. Pääkkönen, C. Hollanti, and O. Tirkkonen, "Device-to-device data storage for mobile cellular systems," in *IEEE Globecom Workshops (GC Wrokshops)*, December 2013, pp. 671–676.

[11] K. Hamidouche, W. Saad, and M. Debbah, "Many-to-many matching games for proactive social-caching in wireless small cell networks," in *WNC3 workshop in conjuction with WiOpt*, Hammamet, Tunisia, May 2014.

[12] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "Match to cache: Optimizing user association and backhaul allocation in cache-aware small cell networks," *[Online] http://goo.gl/OgMGLo*, 2015.

[13] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," *arXiv preprint: 1402.3247*, 2014.

[14] E. Altman, K. Avrachenkov, and J. Goseling, "Coding for caches in the plane," *arXiv preprint: 1309.0604*, 2013.

[15] E. Baştuğ, M. Bennis, M. Kountouris, and M. Debbah, "Cache-enabled small cell networks: Modeling and tradeoffs," *EURASIP Journal on Wireless Communications and Networking*, Accepted (2015).

[16] E. Baştuğ, M. Bennis, and M. Debbah, *Proactive Caching in 5G Small Cell Networks*. Wiley, In Minor Revision (2015).

[17] A. Abboud, R. Couillet, M. Debbah, and H. Siguerdidjane, "Distributed asynchronous optimization of a smart grid network through ADMM," *IEEE Transactions on Signal Processing*, 2015 Submitted.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[19] J. N. Tsitsiklis, D. P. Bertsekas, M. Athans *et al.*, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, June 1986.

[20] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM'99*, vol. 1. IEEE, 1999, pp. 126–134.

[21] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," *arXiv preprint: 1409.7626*, 2014.