Caching Heterogeneous Size Content in Small Cell Networks with CoMP Joint Transmissions

Guilherme Iecker Ricardo[1,2], Giovanni Neglia[2], and Thrasyvoulos Spyropoulos[1]

[1]EURECOM, France, guilherme.ricardo@eurecom.fr, thrasyvoulos.spyropoulos@eurecom.fr
[2]Inria, Université Côte d'Azur, France, giovanni.neglia@inria.fr

***Abstract*—In 5G and beyond network architectures, operators and content providers base their content distribution strategies on small cell networks. On top of such networks, edge caching and Coordinated Multi-Point (CoMP) Joint Transmissions are used to improve performance. Online solutions for average delay minimization problem have been studied in the related literature, although only under the strong assumption that files have equal sizes. In this paper we aim to fill this gap and propose an online caching policy, qLRU-HS, that takes into account heterogeneous sizes and asymptotically converges to the optimal cache allocation under the Independent Reference Model. Our experiments confirm such convergence in practice and reveal that qLRU-HS outperforms other state-of-the-art solutions.**

***Index Terms*—Edge caching, CoMP, joint transmission, heterogeneous cellular networks, optimization, distributed algorithms.**

I. Introduction

Cellular data consumption has experienced an unprecedented increase. According to recent CISCO's forecast [1], by 2023 there will be 13 billion mobile connections, showing an increase of nearly 50% over 2018. Network densification is considered a key strategy to cope with traffic increase [2]. Specifically, 3G/4G macro-cell architecture will be incremented with a large number of overlapping small cells (e.g., femto, pico), in order to improve both coverage and capacity. In a dense cellular network, each user is in general in the transmission range of many BSs and has access to the content of their caches. Cellular networks with this architecture are often call *small cell* networks.

On top of such a densified network, two additional techniques have been proposed to provide higher Quality of Experience (QoE). Assuming that every small base station (BS) has limited data storage capacity, the first technique is *caching* relevant content, e.g., the most popular content (with a higher probability of being requested). It allows users to directly access their desired content from the nearby BSs. As a consequence, the access latency as well as the backhaul congestion and servers' load can be drastically reduced. The second technique is Coordinated Multi-Point (CoMP) *joint transmissions* [3]. The idea is that two or more BSs jointly transmit the requested file to the user. By doing so, users experience higher rates and, consequently, smaller delays to obtain the content. The problem is to define a caching management strategy that is able to optimize the QoE taking CoMP joint transmissions into consideration.

Some related work considers *offline solutions*, where there is a centralized entity aware of the files popularities (assumed to be constant over time) and of the whole network topology [4]–[6]. With this information, it is possible to decide which files should be cached at each BS to optimize a given performance metric, e.g., the probability of finding requested files in the cache, bandwidth usage, etc. However, having all this information available is a very strong assumption and is hardly satisfied in real systems. Moreover, if files have different sizes, the problem complexity increases. Although some works propose interesting solutions to take into account heterogeneous sizes, e.g., through dynamic programming [7] or greedy-based policies [8], the solution is often costly and most of the related literature lacks theoretical bounds and guarantees.

Alternatively, in the *online caching* framework, every BS employs a local caching policy that dynamically updates the local set of files reacting to the request process. Since the BSs take decisions on-the-fly, online policies are more reactive to files' popularity short-term variability [9] and, in comparison to offline solutions, each BS needs to know and exchange much less information. For these reasons, online caching policies are more appropriate to be deployed in real systems. Some related work proposes online caching policies for small cell networks, e.g., to maximize the hit ratio [10]–[12], or minimize the average delay [13], etc. However, these studies are all based on the strong assumption that files have equal size. To the best of our knowledge, online caching policies for different file sizes have been considered only in the single cache setup [14]–[16].

In this paper, we propose an online caching policy that is able to minimize the average delay in a small cell network considering heterogeneous file sizes. In our proposed policy, BSs estimate the marginal gain per byte for keeping a copy of a cached content, calculated as the delay reduction due to the copy divided by the file size. The marginal gain per byte is used to drive the (probabilistic) caching decisions towards the optimal performance.

The main contributions of this paper are summarized as follows:

This work has been supported by the French government, through the EUR DS4H Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-17-EURE-0004 as well as the "5C-for-5G" JCJC project with the reference number ANR-17-CE25-0001.

- In Section II, we present a model that captures file retrieval delay under CoMP joint transmissions opportunities and heterogeneous files sizes.
- We define the offline optimization problem that we use as a baseline for our proposed solution in Section III. To this purpose we consider a greedy algorithm that computes a possibly infeasible caching allocation but with desirable approximation guarantees.
- We introduce our caching policy qLRU-HS in Section IV. qLRU-HS is designed for delay minimization with heterogeneous file sizes and is asymptotically optimal as its parameter q converges to 0.
- In Section V, we provide numerical results based on simulations that show our policy's convergence to the optimum when q vanishes. Then, we evaluate its performance against other policies from the related literature.

II. System Model and Operation

We consider a set $[B]$ of base stations (BSs) arbitrarily located in a given area $A \subseteq \mathbb{R}^2$, where $[n]$ denotes the set $\{1, \ldots, n\}$, for $n \in \mathbb{Z}_+$. There is a set $[U]$ of user equipments (UEs) spread across area A. Because of the high density of BSs, each UE u will, in general, be within communication range of multiple BSs. We denote by I_u the set of UE u's *neighboring* BSs, i.e., all BSs that have UE u within their coverage area and are able to receive requests and transmit content back to u. In order to simplify our analysis, we consider that the Signal-to-Noise Ratio (SNR) $h_u^{(b)}$ of the wireless channel between BS b and UE u is constant, i.e., $h_u^{(b)} = \bar{h} \in \mathbb{R}^+$, if u and b are connected ($b \in I_u$), and $h_u^{(b)} = 0$, otherwise.

Each BS b is equipped with a cache that can store up to $C^{(b)}$ bytes. We consider a catalog of files $[F]$, where file $f \in [F]$ has size equal to s_f bytes and is requested with probability λ_f over the area A (λ_f quantifies then the popularity of content f). In particular we assume that the request process follows the Independent Reference Model (IRM): each request is for file f with probability λ_f, independently from the past. Let $X_f^{(b)} \in \{0,1\}$ be a variable indicating whether BS b caches file f ($X_f^{(b)} = 1$) or not ($X_f^{(b)} = 0$). Then, the vector $\mathbf{X}_f = \left(X_f^{(b)}\right)_{b\in[B]}$ describes the allocation of f across the caches, and the matrix $\mathbf{X} = \left(X_f^{(b)}\right)_{b\in[B], f\in[F]}$ describes the allocation of the entire catalog. Given UE u and allocation $\mathbf{X}_f$, we denote by $J_u(\mathbf{X}_f)$ the set of neighboring BSs of u that are caching f, that is $J_u(\mathbf{X}_f) = \left\{b \in I_u : X_f^{(b)} = 1\right\}$ is actually caching f.

When a UE requires a file, BSs can use CoMP techniques to jointly transmit the file. The *wireless channel access delay* [17], [18] for UE u to download f from k BSs is:

$$d_{u,f}^{\mathrm{WC}}(k) \triangleq \frac{s_f}{w \cdot \log_2\left(1 + \bar{h}\cdot \min(k,\, |I_u|)\right)}, \tag{1}$$

where $w \in \mathbb{R}^+$ is the channel bandwidth and the denominator is the aggregate capacity. The min operator captures the fact that u can download from at most $|I_u|$ neighboring BSs. We consider $d_{u,f}^{\mathrm{WC}}(0) = +\infty$.

The *backhaul-access delay* for any BS to fetch file f from the back-end servers through the backhaul network is:

$$d_f^{\mathrm{BH}} \triangleq r + \frac{s_f}{c^{\mathrm{BH}}}, \tag{2}$$

where c^{BH} is the backhaul network capacity and r is a constant that represents any sort of latency for accessing the back-end servers (e.g., the round-trip time in the backhaul network), henceforth generically referred to as *backhaul latency*.

When UE u wants to retrieve file f, it broadcasts an inquiry message, which is received by u's neighboring BSs in I_u. Then, BSs in I_u estimate the download delay in two cases:

(i) The set of BSs caching f, i.e., $J_u(\mathbf{X}_f)$, directly transmit f to u with delay $d_{u,f}^{\mathrm{WC}}(|J_u(\mathbf{X}_f)|)$.

(ii) A random BS $b' \in I_u$, not caching f, fetches f from the backhaul. Then, BSs in set $J_u(\mathbf{X}_f) \cup \{b'\}$ transmit f to u with delay $d_f^{\mathrm{BH}} + d_{u,f}^{\mathrm{WC}}(|J_u(\mathbf{X}_f)|{+}1)$.

Once the delays are estimated, the BSs proceed to transmit f to u according to the case resulting in the smallest delay.

Therefore, we define the total *end-to-end delay* experienced by UE u to download file f under allocation $\mathbf{X}_f$ as:

$$\begin{aligned} d_{u,f}(|J_u(\mathbf{X}_f)|) \triangleq \min\big(&d_{u,f}^{\mathrm{WC}}(|J_u(\mathbf{X}_f)|), \\ &d_f^{\mathrm{BH}} + d_{u,f}^{\mathrm{WC}}\left(|J_u(\mathbf{X}_f)|{+}1\right)\big). \end{aligned} \tag{3}$$

Note that (3) also captures the delay when *misses* at all caches occur ($J_u(\mathbf{X}_f) = \emptyset$): one neighboring BS b' will fetch the file from the backhaul and transmit it to u.

III. Optimizing Static Cache Allocations

In this section, we consider the static cache allocation problem whose goal is to minimize the average end-to-end delay, assuming that probabilities $\{\lambda_f, \forall f \in [F]\}$ and UEs positions are known. Assuming these quantities are relatively stable over time, the operator could use historical data to estimate them, find the optimal allocation and then prefetch the contents to caches during low-traffic periods of the day, as considered in related works [5], [19].

In particular, if we assume that all UEs are equally likely to generate a request, the delay minimization problem can be formulated as follows:

Problem 1 (Average Delay Minimization Problem)**.**

$$\underset{\mathbf{X}}{\text{minimize}} \quad \bar{d}(\mathbf{X}) \triangleq \sum_{f\in[F]} \lambda_f \frac{1}{U} \sum_{u\in[U]} d_{u,f}(|J_u(\mathbf{X}_f)|) \tag{4}$$

$$\text{subject to} \quad \sum_{f\in[F]} s_f \cdot X_f^{(b)} \le C^{(b)},\ \forall b \in [B], \tag{5}$$

$$X_f^{(b)} \in \{0,1\}, \forall b \in [B], \forall f \in [F]. \tag{6}$$

The objective (4) is the average experienced delay for a request over all files and UEs and $d_{u,f}(\cdot)$ is given by (3). The set of constraints (5) guarantees that any feasible solution meets each BS's cache capacity. Problem 1 is NP-Hard because it is a generalization of the single-cache problem with capacity ("knapsack") constraints, which is NP-Hard [20]. For a general network setting with multiple caches, the problem is NP-hard even in the homogeneous size case [21].

A. Approximate Solution

The following greedy algorithm was introduced in [8, Algorithm 1] to solve the general submodular multiple knapsack problem (SMKP) with a $(1 - 1/e)$ approximation guarantee. Starting from empty caches ($X_f^{(b)} = 0, \forall b, f$), the algorithm iteratively finds the placement (b^*, f^*) that maximizes the ratio between the delay gain and the file size given the current cache allocation $\mathbf{X}$ and adds a copy of f^* to b^*, i.e., it sets $X_{f^*}^{(b^*)} = 1$. Whenever the placement (b^*, f^*) makes b^*'s occupancy reach or exceed its caching capacity, b^* is considered "full" and disregarded in the upcoming iterations. The algorithm stops when all BSs are "full." From now on, we will refer to it as *Infeasible Greedy Algorithm* (IGA), since the resulting allocation is likely to violate constraints (5).

Symmetric instances of Problem 1, where BSs cover equivalent groups of UEs, can be directly mapped to a general instance of SMKP. Therefore, IGA can also be used as an approximate solution in these cases. The objective function (4) was studied in [21] as a set function and the authors proved that it is monotone and submodular for the case where SNRs are homogeneous.[1] Although IGA's solution is likely infeasible and the approximation guarantee is only valid for symmetric setups, it can still be used as a heuristic to approximate the minimum achievable average delay in general instances of Problem 1. We use this approximation as a baseline for the techniques introduced in Section IV. We show in detail in Appendix A how the general solution proposed in [8] can be adapted to Problem 1.

IV. Online Caching Policies

In this section, we assume that there is no centralized intelligence controlling the caching decisions. Instead, BSs manage their cache content on-the-fly, as new requests arrive. Consider that BSs' caches are implemented as ordered queues. Files in the cache are ordered from the most recently used one (at the *front*) to the least recently used one (at the *rear*). The cache can perform three operations: (i) *insert* a new file to the front, (ii) *evict* files from the rear, and (iii) *move-to-the-front* a file already present.

At every request (u, f) from UE u for file f, after u's neighboring BSs serve the file, the respective caches react to that request by performing some of the operations described above. In what follows, we define a variant of qLRU [11] caching policy whose operation depends on the quantity:

$$\frac{\Delta d_{u,f}(k)}{s_f} = \frac{d_{u,f}\,(k-1) - d_{u,f}\,(k)}{s_f}, \tag{7}$$

which is the delay reduction UE u experiences thanks to the k-th copy of file f divided by file f's size (i.e., its delay gain per byte occupied in the cache).

We call our policy qLRU-HS as it is inspired by qLRU and takes explicitly into account files with heterogeneous sizes. We describe qLRU-HS operation as follows:

Upon a request (u, f), given the current allocation $\mathbf{X}_f$:

- All neighboring BSs caching f ($\forall b \in J_u(\mathbf{X}_f)$) independently move f from its current position in the queue to the front with probability:

$$p_{u,f}(|J_u(\mathbf{X}_f)|) \triangleq \beta \cdot \frac{\Delta d_{u,f}(|J_u(\mathbf{X}_f)|)}{s_f}, \tag{8}$$

  where constant β ensures that $p_{u,f}(\cdot) \in (0, 1]$, e.g.,

$$\beta = \min_{u',f',k'>0} \left\{ \frac{s_{f'}}{\Delta d_{u',f'}(k')} \right\}. \tag{9}$$

- For the remaining BSs ($\forall b \in I_u \backslash J_u(\mathbf{X}_f)$): (i) If there is enough cache space, f is directly inserted at the front; (ii) otherwise, with probability q, they evict from the rear enough files to make room for f and insert it to the front. We refer to the file at the rear of the queue as f_{rear}.

We formalize qLRU-HS caching policy in Algorithm 1 from the perspective of each BS b.

Algorithm 1: qLRU-HS Caching Policy (for BS b)

```
Input: $w$, $c^{\text{BH}}$, $r$, $s_f$, $\mathbf{X}_f$, $I_u$, and $\bar{h}$.
if $X_f^{(b)} = 1$ then
    with probability $p_{u,f}(|J_u(\mathbf{X}_f)|)$ in (8) do
        Move-to-the-front $f$
    end
else
    if $C^{(b)} - \sum_{f' \in [F]} s_{f'} \cdot X_{f'}^{(b)} \geq s_f$ then
        Insert $f$ to the front; $X_f^{(b)} \leftarrow 1$
    else
        with probability $q$ do
            while $C^{(b)} - \sum_{f' \in [F]} s_{f'} \cdot X_{f'}^{(b)} < s_f$ do
                Evict file $f_{\text{rear}}$ from the rear; $X_{f_{\text{rear}}}^{(b)} \leftarrow 0$
            end
            Insert $f$ to the front; $X_f^{(b)} \leftarrow 1$
        end
    end
end
```

Remark 1. We note that probability $p_{u,f}(\cdot)$ only depends on (i) sets $J_u(\mathbf{X}_f)$ and I_u, (ii) the file size s_f, and (iii) the average SNR $\bar{h}$. In cellular networks, UEs can measure the SNR of neighboring BSs [22] and piggyback this information in an uplink transmission from u to its neighboring BSs, with negligible overhead.[2]

When each cache deploys qLRU-HS, the whole network's cache allocation probabilistically changes with time as new requests arrive. Under the Characteristic Time Approximation

[1] Note that, as the guarantee holds for the maximization of a non-decreasing submodular function, we need to transform the minimization in Problem 1 in an equivalent maximization problem.

[2] The setting (9) would require each BS to be aware of the sizes of all files in the catalog. To avoid this issue in practice, every BS can estimate β on-the-fly, based on previous requests.

(CTA) [23], [24], and the Exponentialization Approximation (EA) [10], we can represent such process, as a set of coupled Continuous-Time Markov Chains (MCs). When q tends to 0, these MCs admit stationary distributions, which, in turn, correspond to the optimal solution of the continuous relaxation of Problem 1. Therefore, although BSs run the qLRU-HS policy individually, they implicitly coordinate to achieve the optimal cache allocation. This result is formalized as follows:

Proposition 1. *Under IRM, CTA, and EA, a network of* qLRU-HS *caches asymptotically achieves an optimal caching configuration, when* $q \to 0$, *even if files have different sizes.*

Proposition 1 is based on [25, Prop. IV.1], which states the optimality of another policy for the case where all files have the same size. Due to space limitations, we present the detailed proof of Proposition 1 in Appendix B. Here, we provide an intuitive explanation of why optimality holds.

Intuition: We observe that, as q converges to 0, the cache exhibits two different dynamics: The *insertion* of new files tends to happen more and more rarely (q converges to 0), while the frequency of *moves-to-the-front* for files already in the cache is unchanged ($p_{u,f}(\cdot)$ does not depend on q). A file f at cache b is moved to the front with a probability proportional to the placement's cost-benefit $\Delta d_{u,f}(|J_u(\mathbf{X}_f)|)/s_f$, i.e., (i) proportional to how much the file contributes to reducing the delay of that specific request and (ii) inversely proportional to how much cache space it takes. The expected number of moves-to-the-front file f experiences depends on (i) how often it is requested (λ_f) and (ii) how likely it is to be moved to the front upon a request ($p_{u,f}(\cdot)$). By the law of large numbers, the random number of moves-to-the-front will be close to its expected value and the least valuable file in the cache likely occupies the last position. We can then think that, when a new file is inserted in the cache, it will replace files that contribute the least to the decrease of the expected cost. qLRU-HS progressively replaces the least useful files from the cache, until it reaches a global minimum.

V. Numerical Results

In this section, we first study qLRU-HS convergence to the optimal cache allocation when q tends to 0 and then we evaluate its performance in different scenarios by comparing it against other policies from related literature, including:

- qLRU-Δd [13], it aims to minimize the average delay in a small-cell CoMP-aware setup, but considers that all files have the same size.
- *greedy-dual-size* [15], it aims to maximize the hit ratio in a single-cache setup, considering sizes are heterogeneous. We consider that *all* BSs run an instance of greedy-dual-size and react independently to each request in their cell. We refer to such operation as GDSize-All in analogy to multi-LRU-All in [12].
- IGA greedy algorithm [8], as discussed in Section III, its average delay reduction is guaranteed to be $(1-1/e)$ far from the optimal. Thus, we use it as a baseline for the other policies.

In our experiments, we consider the *Berlin topology*: a cellular network consisting of $B = 10$ BSs located according to the positions of T-mobile BSs in Berlin extracted from [26]. We call *network density*, ρ, the average number of BSs covering a UE and we assume that UEs are homogeneously distributed within the BSs' coverage area. In this network, all BSs have the same cache capacity, i.e., $C^{(b)} = C, \forall b \in [B]$, and can store up to $C = 50$ GB. Unless otherwise specified, we consider that the backhaul network is able to transmit data at $c^{\text{BH}} = 100$ Mbps with backhaul latency $r = 10$ ms. The wireless channel bandwidth is $w = 5$ MHz and all connected pairs BS-UE have average SNR of $\bar{h} = 10$ dB. All these values are consistent with related literature [6], [25].

In our simulations, we consider that, at every request, a file is chosen from a catalog of $F = 10^4$ files with probability determined by a Zipf law with exponent $\alpha = 0.8$. As suggested by [7], real file sizes may be represented by a truncated exponential distribution. We randomly generate the file sizes according to an exponential distribution within the interval $[s_{\min}, s_{\min} + \Delta s]$. Unless otherwise specified, we consider $s_{\min} = 1$ GB and $\Delta s =$ 9GB. We split the simulation into warm-up and measurement phases, each having 10^7 requests.

A. Convergence Analysis

According to Proposition 1, as q tends to 0, qLRU-HS converges to an optimal allocation. In our first experiments, our goal is to observe this convergence in practice. We consider the Berlin topology with density of $\rho = 5.9$ BSs/UE.

In Fig. 1, we show the average delay (left) and the hit ratio (right) versus the parameter q. As a reference, we include the the result of IGA for the same setup, which is independent of parameter q. We emphasize that, although IGA may be unfeasible, its delay saving is not farther than $(1-1/e)$ from the optimal. As we observe in Fig. 1 (left), qLRU-HS gets closer to IGA as q decreases, suggesting its convergence to the optimal allocation. In addition to qLRU-HS results, we also plot the results for qLRU-Δd, that is also guaranteed to converge to the minimum delay as q vanishes, but only when files have all the same size [21]. However, qLRU-Δd converges to a value of average delay larger than qLRU-HS's one. This is due to fact that qLRU-Δd, while trying to minimize the delay, tends to store large files, that indeed incur large transmission delay, ignoring that they also occupy a large amount of space in the cache. In particular, given two files f_1 and f_2 with $\lambda_{f_1} > \lambda_{f_2}$ and $s_{f_2} \gg s_{f_1}$, qLRU-Δd would prefer f_2, while our caching policy qLRU-HS correctly bias its choices in favor of f_1 that leads to a larger benefit for byte occupied in the cache. From Fig. 1 (right), we see that, for this particular scenario, better average delay is associated with a better hit ratio, which is not always necessarily the case.

In Fig. 2, we show the average delay (left) and the hit ratio (right) versus the number of requests in the simulation. For this plot, we simulate qLRU-HS and qLRU-Δd for $q = 10^{-3}$ and $q = 10^{-4}$, and we indicate the results of IGA as reference. As we observe in Fig. 2 (left), the average delay achieved by each policy decreases over time, and reaches its minimum value after about 10^6 requests (10^5 requests per BS).

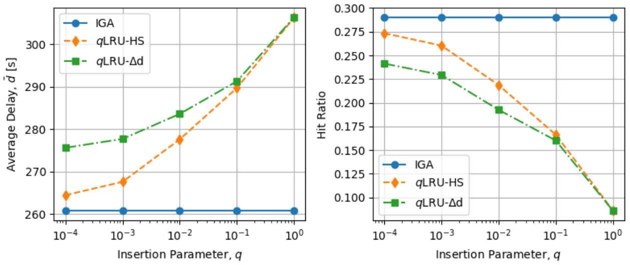


Fig. 1: Average delay $\bar{d}$ (left) and hit ratio (right) versus q. $C = 50.0$ GB, $\rho = 5.9$ BSs/UE, $r = 10.0$ ms, $s_{\min} = 1.0$ GB, and $\Delta s = 9.0$ GB.

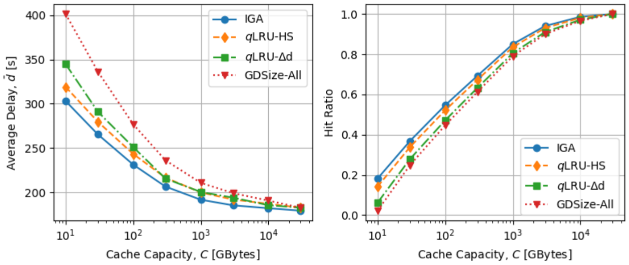


Fig. 3: Average delay $\bar{d}$ (left) and hit ratio (right) versus cache capacity C. $\rho = 5.9$ BSs/UE, $s_{\min} = 1$ GB, $q = 10^{-3}$, and $\Delta s = 9$ GB.

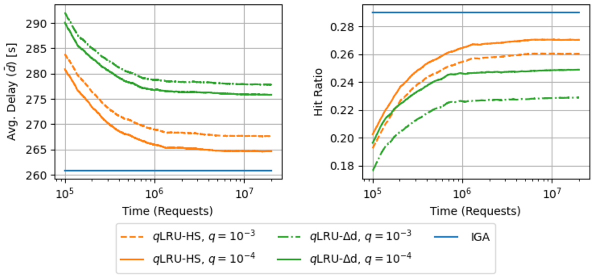


Fig. 2: Average delay $\bar{d}$ (left) and hit ratio (right) versus number of requests. Results of qLRU-HS and qLRU-Δd are shown for $q = 10^{-3}$ and $q = 10^{-4}$.

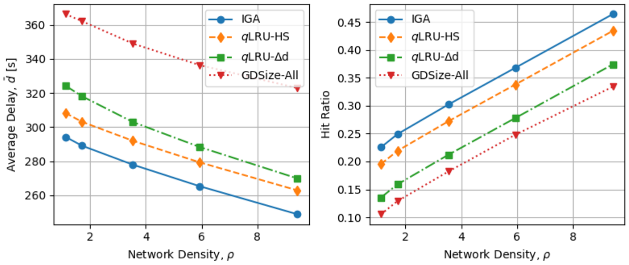


Fig. 4: Average delay $\bar{d}$ (left) and hit ratio (right) versus network density ρ. $C = 30$GB, $s_{\min} = 1$ GB, $\Delta s = 9$ GB, and $q = 10^{-3}$.

B. Performance Evaluation

Now, we compare the performance of qLRU-HS with other caching solutions in different scenarios. From now on, we consider $q = 10^{-3}$ for qLRU-HS and qLRU-Δd.

In Fig. 3, we show the performance for different values of caching capacity, ranging from $C = 10$ GB to $C = 100$ TB. We present the average delay (left) and the hit ratio (right) versus the cache capacity size C. qLRU-HS provides a more efficient management of the cache, outperforming all other policies and presenting results close to the IGA ones. The difference of performance across policies is maximal for smaller values of C, in particular. for $C = 10$ GB. qLRU-HS achieves a delay about 20% smaller than GDSIZE-ALL. As expected, when the capacity increases, all policies perform better because they can store more files and also differences reduce until all policies perform equally when the cache is so large to be able to store the whole catalog.

In Fig. 4, we fix the cache capacity to $C = 30$ GB and observe the policies' performances for different levels of density, from $\rho = 1.4$ BSs/UE to $\rho = 9.1$ BSs/UE. We control the network density by simply increasing the BSs' transmission range, although we keep constant the SNR to $h = 10$ dB. In this scenario, qLRU-HS again outperforms all other policies and has results close to the IGA ones.

We observe in Fig. 4 (left) that all policies experience a delay reduction as ρ increases. The reason is that the aggregate cache available to each UE gets larger with ρ, then more files are found in the neighboring caches. Because of the larger aggregate cache, also the difference between qLRU-HS and qLRU-Δd becomes slightly smaller as ρ increases (similarly to what observed in Fig. 3). On the contrary the performance gap with GDSIZE-ALL increases: the fact that all BSs in I_u react to a request from u leads to poor coordination.

Fig. 5 shows the average delay $\bar{d}_{\text{P}}$ achieved by policy P normalized by the average delay $\bar{d}_{\text{IGA}}$ achieved by IGA. Results are presented for different size variability (captured by the parameter Δs), on the left, and backhaul latency r, on the right. For these experiments, we fix the network density to $\rho = 5.9$ BSs/UE. We chose to show the results in a normalized fashion due to the large excursion of $\bar{d}_{\text{P}}$ values when both Δs and r change.

In Fig. 5 (left) we evaluate $\bar{d}_{\text{P}}/\bar{d}_{\text{IGA}}$ for fixed $s_{\min} = 1$ GB and change the Δs from $\Delta s = 0$ (homogeneous file sizes) to $\Delta s = 49$ GB. We first observe that qLRU-HS and qLRU-Δd both have results close to IGA in the homogeneous size case. The more heterogeneous is the catalog, in terms of size, the more noisy is the convergence process, as the insertion of a single large file can lead to the eviction of many other files and significantly change the quality of the current allocation. This fact explains why the relative performance of all dynamic policies worsens when size variability increases. Despite the increasing trend shared by all policies, we observe that qLRU-HS is always the closest to IGA. Interestingly, although GDSIZE-ALL has the worst performance, it is less sensitive to the variability of file sizes.

Finally, one interesting aspect in our model is how the

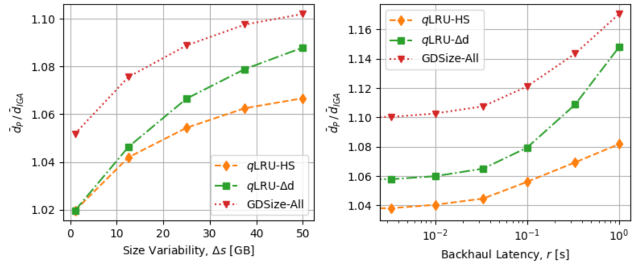


Fig. 5: $\bar{d}_{\text{P}}/\bar{d}_{\text{IGA}}$ by size variability Δs (left) and backhaul latency r (right). $C = 30.0$ GB, $q = 10^{-3}$, and $\rho = 5.9$ BSs/UE.

backhaul latency constant affects the policies operation and results. In Fig. 5 (right), we show $\bar{d}_{\text{P}}/\bar{d}_{\text{IGA}}$ when the backhaul latency increases from $r = 30$ ms to $r = 1$ s. In this case, we fixed $s_{\min} = 1$ MB and the size variability to $\Delta s = 9.0$ MB. In this experiment, we also observe dynamic policies perform worse in comparison to IGA as the backhaul latency r increases. When r becomes larger, the optimal caching strategy changes from a scenario where it is convenient to store more copies of the same files across the BSs' caches (to create CoMP opportunities) to a scenario where file diversity across caches is preferred because it minimizes cache misses that cause the largest delay. This means that, for large enough values of backhaul latency, qLRU-Δd and qLRU-HS take an equivalent strategy, to diversify files throughout the network of caches. However, qLRU-Δd still erroneously prefer to store large files. This leads to qLRU-Δd storing on average less files, which decreases the hit probability and, in turn, worsens qLRU-Δd's performance. On the contrary, GDSize-All correctly prefer the smallest files, but, as all caches react at the same time, BSs tend to have similar cache content. This replication of files throughout the BSs is suboptimal for high latency, which explains GDSize-All's worse performance.

VI. Conclusion

In this paper we proposed an online caching policy for average delay minimization in a small cell architecture with CoMP joint transmissions and heterogeneous file sizes. We formulated the static optimization problem for which an infeasible greedy algorithm provides approximation guarantees in the homogeneous SNR regime. Then, we introduced a novel online caching policy able to converge to the optimal caching allocation that minimizes the delay under IRM. In our experiments, we observed qLRU-HS's convergence and evaluated its performance under different request processes and SNR regimes. We conclude that qLRU-HS achieves considerable performance gains with negligible additional deployment complexity.

References

[1] CISCO, "Cisco annual internet report (2018–2023)," CISCO, Tech. Rep., March 2020.

[2] N. Bhushan *et al.*, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, Feb. 2014.

[3] D. Lee, H. Seo, B. Clerckx, E. Hardouin, D. Mazzarese, S. Nagata, and K. Sayana, "Coordinated multipoint transmission and reception in LTE-advanced: deployment scenarios and operational challenges," *IEEE Communications Magazine*, vol. 50, no. 2, pp. 148–155, Feb. 2012.

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

[5] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, Dec 2013.

[6] A. Tuholukova, G. Neglia, and T. Spyropoulos, "Optimal cache allocation for Femto helpers with joint transmission capabilities," in *IEEE ICC 2017, 21-25 May 2017, Paris, France*, Paris, France, 05 2017.

[7] T. Mihretu Ayenew, D. Xenakis, N. Passas, and L. Merakos, "A novel content placement strategy for heterogeneous cellular networks with small cells," *IEEE Networking Letters*, vol. 2, no. 1, pp. 10–13, 2020.

[8] X. Sun, J. Zhang, and Z. Zhang, "Deterministic algorithms for the submodular multiple knapsack problem," 2020, arXiv:2003.11450.

[9] S. Traverso *et al.*, "Temporal Locality in Today's Content Caching: Why It Matters and How to Model It," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, Nov. 2013.

[10] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1276–1285, June 2018.

[11] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 3, pp. 12:1–12:28, May 2016.

[12] A. Giovanidis and A. Avranas, "Spatial multi-lru caching for wireless networks with coverage overlaps," *SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 403–405, Jun. 2016.

[13] G. Ricardo, G. Neglia, and T. Spyropoulos, "Caching policies for delay minimization in small cell networks with joint transmissions," in *IEEE ICC 2020*, Dublin, Ireland, 06 2020.

[14] G. Neglia, D. Carra, M. Feng, V. Janardhan, P. Michiardi, and D. Tsigkari, "Access-time-aware cache algorithms," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 4, pp. 21:1–21:29, Nov. 2017.

[15] Shudong Jin and A. Bestavros, "Popularity-aware greedy dual-size web proxy caching algorithms," in *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, 2000, pp. 254–261.

[16] D. S. Berger, R. K. Sitaraman, and M. Harchol-Balter, "Adaptsize: Orchestrating the hot object memory cache in a content delivery network," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 483–498.

[17] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[18] W. C. Ao and K. Psounis, "Distributed caching and small cell cooperation for fast content delivery," in *MobiHoc*. ACM, 2015, pp. 127–136.

[19] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3665–3677, Oct 2014.

[20] M. Chrobak, G. J. Woeginger, K. Makino, and H. Xu, "Caching is hard—even in the fault model," *Algorithmica*, vol. 63, no. 4, pp. 781–794, 2012.

[21] G. I. Ricardo, A. Tuholukova, G. Neglia, and T. Spyropoulos, "Caching policies for delay minimization in small cell networks with coordinated multi-point joint transmissions," *IEEE/ACM Transactions on Networking*, to appear.

[22] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2011.

[23] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: modeling, design and experimental results," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 7, pp. 1305–1314, Sep 2002.

[24] R. Fagin, "Asymptotic miss ratios over independent references," *Journal of Computer and System Sciences*, vol. 14, no. 2, pp. 222 – 250, 1977.

[25] G. Neglia, E. Leonardi, G. I. Ricardo, and T. Spyropoulos, "A Swiss Army Knife for Dynamic Caching in Small Cell Networks," 2020, arXiv:1912.10149.

[26] "Openmobilenetwork." [Online]. Available: openmobilenetwork.org/

Supplementary Material for paper:

Caching Heterogeneous Size Content in Small Cell Networks with CoMP Joint Transmissions

Appendix A
Notes on IGA

A. Submodular Multiple Knapsack Problem (SMKP)

Let $f : 2^V \to \mathbb{R}_+$ be a non-negative, monotone, and submodular set function defined over a generic ground set V. Let also $w : 2^V \to \mathbb{R}_+$ be the weight function defined for any subset of V. For singleton sets, the weight function represents the storage cost of that particular element. In this case, we abuse the notation and denote the element weight as $w(\{v\}) = w(v)$.

There is a set $K = \{1, 2, \ldots, K\}$ of knapsacks (or bins), such that each knapsack $k \in [K]$ has total storage capacity $C^{(k)}$. We denote the solution set by $S \subseteq V$ and it may be partitioned into K disjoint subsets, i.e., $S = (S^{(1)}, \ldots, S^{(K)})$, indicating the placement at every knapsack. For any feasible solution S, its elements may be arbitrarily placed into the available bins as long as the knapsack capacity constraints are satisfied, i.e.,

$$w\left(S^{(k)}\right) = \sum_{v \in S^{(k)}} w(v) \leq C^{(k)}, \forall k \in [K]. \tag{10}$$

The goal of the Submodular Multiple Knapsack Problem (SMKP) is to find the optimal solution set $S_{\text{OPT}} \subseteq V$ that is feasible and maximizes the objective function $f(\cdot)$. We formalize SMKP as follows:

Problem 2 (Submodular Multiple Knapsack Problem – SMKP)**.**

$$\begin{aligned} \underset{S \subseteq V}{\text{maximize}} \quad & f(S) \\ \text{subject to} \quad & w(S^{(k)}) \leq C^{(k)}, \forall k \in [K]. \end{aligned}$$

We emphasize some important notes about SMKP:

1) The elements in a solution contribute to the objective function regardless of which knapsack they are placed at.
2) The same element cannot be placed at multiple knapsacks. By construction, replication represents no change to the objective.

SMKP is an NP-Hard problem, so we present IGA (Algorithm 2) as an approximate solution, as introduced in [8]. In the submodular optimization context, we define the objective's discrete derivative as:

$$\Delta f(v|S) \triangleq f(S \cup \{v\}) - f(S), \tag{11}$$

which is the marginal gain for adding element v to the current solution S. Note that, in line 3 of Algorithm 2, IGA considers the marginal gain of a placement (its discrete derivative) relative to its weight (or its storage cost), as a way to reflect the placement's cost-benefit. We emphasize that, although IGA enjoys a $(1 - 1/e)$-approximation guarantee (the proof is in [8]), it is likely to provide an infeasible solution (the solution may violated the knapsack capacity constraints).

Algorithm 2: IGA algorithm for general SMKP

```
  input : Sets $V$ and $[K]$; Functions $f(\cdot)$ and $w(\cdot)$
  output: Solution set $S$ and how it is partitioned.
1 $S \leftarrow \emptyset$
2 while $\exists k \in [K] : w(S^{(k)}) \leq C^{(k)}$ do
3     $v^* \leftarrow \underset{v' \in V \setminus S}{\arg\max} \left\{ \frac{\Delta f(v'|S)}{w(v')} \right\}$
4     $S^{(k)} \leftarrow S^{(k)} \cup \{v^*\}$
5 end
6 return $S = \left(S^{(1)}, \ldots, S^{(K)}\right)$
```

B. Mapping Problem 1 to SMKP

First, we assume a symmetric setup for Problem 1, i.e., BSs are symmetrically located so that they cover an equivalent number of UEs and operate under homogeneous conditions (e.g., same SNRs). In what follows, for the sake of ease of presentation, we consider the simplest symmetric setup, where all B BSs completely overlap.

Let $V = \{v_{f,k} : \forall [F] \times [B]\}$ be the ground set, where element $v_{f,k}$ represents the k-th copy of file f in the cache network. We consider a set of bins (knapsacks) $[B]$, where each bin $b \in [B]$ has capacity $C^{(b)}$.

We define the weight function $w : 2^V \to \mathbb{R}_+$, such that the weight of subset $X \subseteq V$ is:

$$w(S) \triangleq \sum_{v_{f,k} \in S} s_f, \tag{12}$$

where s_f is the size of the file associated to element $v_{f,k}$. For singleton sets, the weight function represents the storage cost of that particular element. In this case, we abuse the notation and denote the element weight as $w(\{v_{f,k}\}) = w(v_{f,k}) = s_f$.

We represent the solution set by $X \subseteq V$, which is partitioned according to the set of bins, i.e., $X = (X^{(1)}, \ldots, X^{(B)})$. We make a parallel between the set notation introduced here with the matrix notation used to described the system in Section II: element $v_{f,k}$ is in solution X if, in the current allocation $\mathbf{X}$, $\sum_{b \in [B]} X_f^b = k$. For any feasible solution X, its elements may be arbitrarily placed into the available bins as long as the knapsack capacity constraints are satisfied, i.e.,

$$w\left(S^{(b)}\right) = \sum_{v_{f,k} \in S^{(b)}} s_f \le C^{(b)}, \forall b \in [B]. \tag{13}$$

Now, we define a profit function $d : V \to \mathbb{R}_+$ as follows:

$$d(v_{f,k}) \triangleq \lambda_f \frac{1}{U} \sum_{u \in [U]} \left(d_{u,f}(k-1) - d_{u,f}(k)\right) = \lambda_f \left(d_f(k-1) - d_f(k)\right), \tag{14}$$

where we can just drop references to multiple to UEs, such that $d_f(k) = d_{u,f}(k), \forall u \in [U]$ and consider a single UE due to the setup symmetry. Then, the goal is to maximize the total profit of allocation X, satisfying knapsack capacity constraints. We formalize the new optimization problem as follows:

Problem 3 (Delay Reduction Maximization – DRMax)**.**

$$\begin{aligned} \underset{X \subseteq V}{\text{maximize}} \quad & D(X) \triangleq \sum_{v \in X} d(v) \\ \text{subject to} \quad & w(X^{(b)}) \le C^{(b)}, \forall b \in [B], \end{aligned} \tag{15}$$

where $D : 2^V \to \mathbb{R}_+$ is the total profit of all elements in $X \subseteq V$. Note that minimizing (4) is equivalent to maximizing (15).

Finally, we can define the objective function's discrete derivative as follows:

$$\begin{aligned} D(v_{f,k} \,|\, X) \triangleq D(X \cup \{v_{f,k}\}) - D(X) &= \sum_{v \in X \cup \{v_{f,k}\}} d(v) - \sum_{v \in X} d(v) \\ &= d(v_{f,k}) = \lambda_f \left(d_f(k-1) - d_f(k)\right). \end{aligned}$$

This means that the marginal profit gain is actually the delay reduction of adding an extra copy of file f to the final solution (at any BS). We also note that, in this symmetric setup, the marginal profit gain becomes independent of the current allocation

The following greedy algorithm can be used to approximate Problem 3 and it is an adapted version of Algorithm 2: Starting from an empty solution ($X \leftarrow \emptyset$), at every iteration, the algorithm finds the the element $v^* \in V \setminus X$ that maximizes the delay reduction $\frac{D(v|X)}{w(v)}$ related to its storage cost and adds it to the current solution X at any available bin. Whenever the element v^* makes a bin's occupancy reach or exceed its capacity, that bin is considered "full" and disregarded in the upcoming iterations. The algorithm stops when all bins are "full." This is IGA adapted to Problem 3 and we formalize it in Algorithm 3.

We emphasize that that, by construction (see definition (14), elements are added to the final solution in order, i.e., first $v_{f,1}$, then $v_{f,2}$, etc. This means that, when the algorithm adds the k-th copy of a file, it had already added all $1, \ldots, k-1$ copies, necessarily. Moreover, in this symmetric setup, it does not make a difference at which BS files are being placed at (or even if there is multiple copies of the same file at the same BS). The whole system works a single cache with expanded capacity $\sum_{b \in [B]} C^{(b)}$. Therefore, for the symmetric setup, IGA may be used to solve Problem 3 with $(1 - 1/e)$-approximation guarantee. In general setups, although IGA no longer enjoys such optimality guarantee, it can still be used as an upper bound (or a lower bound, if we consider Problem 1.)

Algorithm 3: IGA for (Symmetric) DRMax

```
input : Sets: $[U]$, $[F]$, and $[B]$; Parameters: $w$, $c^{\mathrm{BH}}$, $r$, $\lambda_f, s_f, \forall f \in [F]$, $C^{(b)}, \forall b \in [B]$, $I_u, \forall u \in [U]$, and
        $h_u^{(b)}, \forall b \in [B], \forall u \in [U]$; and Functions: $D(\cdot)$ and $w(\cdot)$.
output: Solution set $X$ and it is partitioned.
$X \leftarrow \emptyset$
while $\exists b \in [B] : w(S^{(b)}) \leq C^{(b)}$ do
    $v^* \leftarrow \underset{v' \in V \setminus X}{\arg\max} \left\{ \frac{\Delta D(v' | X)}{w(v')} \right\}$
    $X^{(b)} \leftarrow X^{(b)} \cup \{v^*\}$
end
return $X = \left( X^{(1)}, \ldots, X^{(B)} \right)$
```

Appendix B
qLRU-HS Optimality Proof

Consider the exponential-size linear reduction of Problem 1:

Problem 4 (ADMin Exponential-Size Linear Reduction)**.**

$$\underset{\mathbf{Y}}{\text{minimize}} \quad \sum_{f \in [F]} \sum_{\mathbf{x}_f} Y_f(\mathbf{x}_f) \cdot D_f(\mathbf{x}_f) \tag{16}$$

$$\text{subject to} \quad \sum_{f \in [F]} \sum_{\mathbf{x}_f} s_f \cdot Y_f(\mathbf{x}_f) \cdot x_f^{(b)} \leq C^{(b)}, \forall b \in [B], \tag{17}$$

$$\sum_{\mathbf{x}_f} Y_f(\mathbf{x}_f) = 1, \forall f \in [F], \tag{18}$$

where $D_f(\mathbf{x}_f) \triangleq \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} d_{u,f}(|J_u(\mathbf{x}_f)|)$.

In Problem 4, we introduce a new set of variables $Y_f(\mathbf{x}_f) \in \{0, 1\}, \forall f \in [F], \forall \mathbf{x}_f \in \{0, 1\}^B$ that indicate whether, for file f, the allocation $\mathbf{x}_f$ is considered in the solution. Then, objective (16) is equivalent to (4). Constraints (17) are the cache capacity constraints adapted to the new variables $Y_f(\mathbf{x}_f)$. Finally, we introduce a new set of constraints (18) to guarantee that only one assignment $\mathbf{x}_f$ is considered in the final solution for each file f. Therefore, Problem 1 and Problem 4 are equivalent.

A. Assumptions and Notation

In the framework of Section IV, the content of caches change over time as new requests arrive, sporadically causing new insertions and moves-to-the-front. These operations are based on probabilities, so we introduce the random variable $\mathbf{X}_f(t)$ to represent the allocation of file f across the network of caches at time t. Any actual assignment of $\mathbf{X}_f(t)$ can be generically expressed by $\mathbf{x}_f \in \{0, 1\}^B$. Therefore, $\{\mathbf{X}_f(t) : t \geq 0\}$ (or simply $\mathbf{X}_f(t)$) is the stochastic process representing the evolution of file f's allocation across the network of qLRU-HS caches over time.

Under Characteristic Time Approximation (CTA) the instant cache capacity constraints (5) can be violated. Instead, any file inserted at BS b has an associated timer with fixed duration $T_c^{(b)}$, the characteristic time. Only by the end of $T_c^{(b)}$, the file is evicted from b's cache. Moreover, the timer is reset upon every subsequent cache hit, such that the expected sojourn time of file f at BS b's cache in a given starting cache configuration $\mathbf{x}_f$ is:

$$\mathbb{E}[T_{S,f}^{(b)}(\mathbf{X}_f(t))] = \frac{e^{\beta \cdot \Delta D_f^{(b)}(\mathbf{X}_f(t)) \cdot T_c} - 1}{\beta \cdot \Delta D_f^{(b)}(\mathbf{X}_f(t))} = \frac{1}{\nu_f^{(b)}}, \tag{19}$$

where $\Delta D_f^{(b)}(\mathbf{x}_f) = D_f(\mathbf{x}_f \ominus \mathbf{e}^{(b)}) - D_f(\mathbf{x}_f)$ is the delay gain for keeping file f at BS b's cache and $\nu_f^{(b)}$ is the rate at which f is evicted from b.

Second, we consider the Exponentialization Approximation (EA), where the stochastic process $\mathbf{X}_f(t)$ can be simplified by a Continuous-Time Markov Chain (CTMC). EA treats the dynamics of each file individually and independently of other files. Lastly, we note that the CTMC representation of the heterogeneous size content case is analogous to the homogeneous size content case, as originally proposed in [25]. In Figure 6, we show an example of a CTMC for a given file f and a scenario with $B = 2$ BSs. The set of states consists of each possible assignment of $\mathbf{X}_f(t)$, in this case $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$, and any transition from state $\mathbf{x}_f$ to $\mathbf{y}_f$ has rate $\rho[\mathbf{x}_f \rightarrow \mathbf{y}_f]$.

Consider two states $\mathbf{x}_f, \mathbf{y}_f$, such that $\|\mathbf{y}_f\| > \|\mathbf{x}_f\|$. There are two types of transitions:

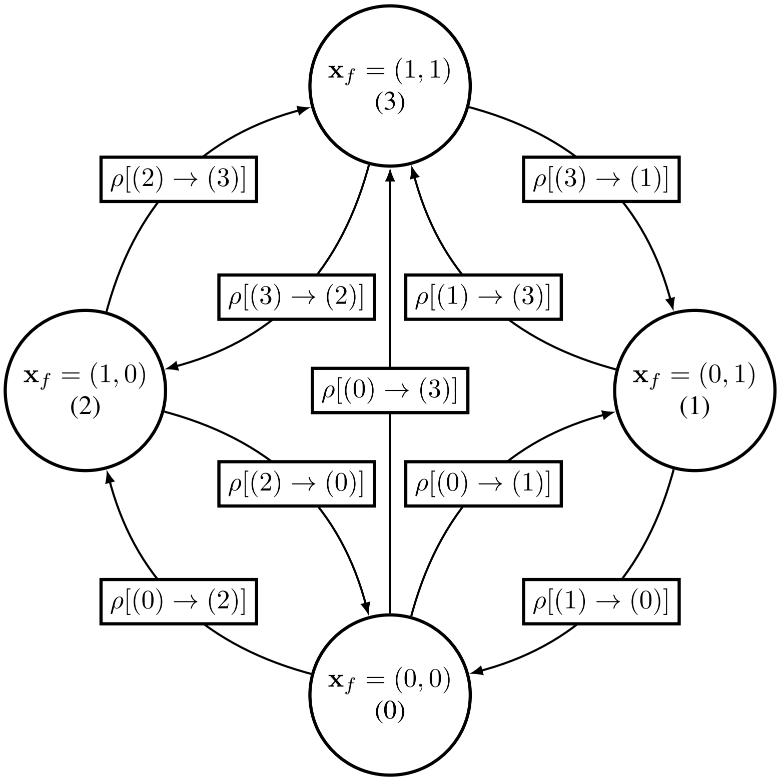


Fig. 6: CTMC $\mathbf{X}_f(t)$ for $B = 2$ BSs.

- Upward transitions with rate $\rho[\mathbf{x}_f \to \mathbf{y}_f] \propto q^{s_f \gamma^\top (\mathbf{y}_f - \mathbf{x}_f)}$
- Downward transitions with rate $\rho[\mathbf{y}_f \to \mathbf{x}_f] \propto q^{\Delta G_f^{(b_0)}(\mathbf{y}_f)}$, where $\mathbf{x}_f = \mathbf{y}_f \ominus \mathbf{e}^{(b_0)}$.

The *direct resistance* $r_f(\mathbf{x}_f, \mathbf{x}'_f)$ of transition $\mathbf{x}_f \to \mathbf{x}'_f$ is defined as the exponent of parameter q, i.e., upward transitions have resistance $r_f(\mathbf{x}_f, \mathbf{y}_f) = s_f \gamma^\top (\mathbf{y}_f - \mathbf{x}_f)$ and downward transitions have resistance $r_f(\mathbf{y}_f, \mathbf{x}_f) = \Delta G_f^{(b_0)}(\mathbf{y}_f)$.

By sampling the CTMC $\mathbf{X}_f(t)$ every τ time units, we obtain the corresponding DTMC $\hat{\mathbf{X}}_f(k) = \mathbf{X}_f(k\tau)$. The transition probability matrix $\mathbf{P}_{f,q}$ is defined for every value of parameter q. The probability of transition $\mathbf{x}_f \to \mathbf{x}'_f$ is $P_{f,q}(\mathbf{x}_f, \mathbf{x}'_f) \propto q^{r_f(\mathbf{x}_f, \mathbf{x}'_f)}$.

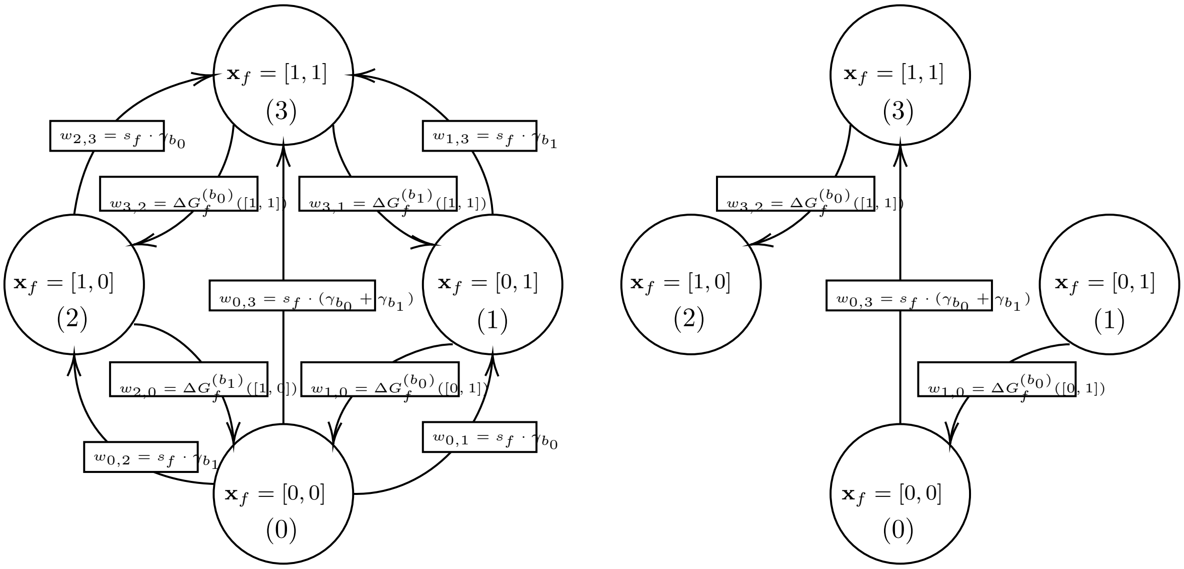
Let $\mathcal{G}_f$ be a weighted, directed graph, where nodes are all states $\mathbf{x}_f \in \{0,1\}^B$ and each possible transition of $\hat{\mathbf{X}}_f(k)$ is mapped to an edge with weight equal to the corresponding transition's direct resistance. For example, for $B = 2$, $\mathcal{G}_f$ is represented in Figure (7a). Let $\mathcal{T}(\mathbf{x}_f)$ be an in-tree on $\mathcal{G}_f$ rooted to $\mathbf{x}_f$ containing all nodes and $\mathcal{I}(\mathbf{x}_f)$ be the set of all possible in-trees built this way. Figure (7b) shows an example of in-tree over $\mathcal{G}_f$. The resistance $r_f(\mathcal{T}(\mathbf{x}_f))$ of in-tree $\mathcal{T}(\mathbf{x}_f)$ is the sum of the resistances of the transitions composing it, i.e., $r_f(\mathcal{T}(\mathbf{x}_f)) \triangleq \sum_{(\mathbf{y}_f, \mathbf{y}'_f) \in \mathcal{T}(\mathbf{x}_f)} r_f(\mathbf{y}_f, \mathbf{y}'_f)$.

Definition 1: The *resistance* $r_f(\mathbf{x}_f)$ of state $\mathbf{x}_f$ is $r_f(\mathbf{x}_f) \triangleq \min_{\mathcal{T} \in \mathcal{I}(\mathbf{x}_f)} r_f(\mathcal{T})$.

For $q \to 0$, $\hat{\mathbf{X}}_f(k)$ is irreducible, aperiodic, and finite. Thus, $\hat{\mathbf{X}}_f(k)$ admits stationary probabilities, which are denoted by vector $\boldsymbol{\pi}_{f,q} = \{\pi_{f,q}(\mathbf{x}_f), \forall \mathbf{x}_f \in \{0,1\}^B\}$.

Definition 2: If $\lim_{q \to 0} \pi_{f,q}(\mathbf{x}_f) > 0$, then $\mathbf{x}_f$ is called a *stochastically stable state*.

Property 1. *By [25, Lemma IV.3], $\pi_{f,q}(\mathbf{x}_f) \propto q^{r_f(\mathbf{x}_f) - \min_{\mathbf{x}'_f \in \{0,1\}^B} r_f(\mathbf{x}'_f)}$, i.e., stochastically stable states are those with minimal resistance. This happens because $q \to 0$ and any exponent different from 0 will result in probability 0 (as well as any exponent equals 0, will result in a probability proportional to 1.)*


$\mathbf{x}_f = [1,1]$
(3)
$w_{2,3} = s_f \cdot \gamma_{b_0}$
$w_{1,3} = s_f \cdot \gamma_{b_1}$
$w_{3,2} = \Delta G_f^{(b_0)}([1,1])$
$w_{3,1} = \Delta G_f^{(b_1)}([1,1])$
$\mathbf{x}_f = [1,0]$
(2)
$w_{0,3} = s_f \cdot (\gamma_{b_0} + \gamma_{b_1})$
$\mathbf{x}_f = [0,1]$
(1)
$w_{2,0} = \Delta G_f^{(b_1)}([1,0])$
$w_{1,0} = \Delta G_f^{(b_0)}([0,1])$
$w_{0,1} = s_f \cdot \gamma_{b_0}$
$w_{0,2} = s_f \cdot \gamma_{b_1}$
$\mathbf{x}_f = [0,0]$
(0)
$\mathbf{x}_f = [1,1]$
(3)
$w_{3,2} = \Delta G_f^{(b_0)}([1,1])$
$\mathbf{x}_f = [1,0]$
(2)
$w_{0,3} = s_f \cdot (\gamma_{b_0} + \gamma_{b_1})$
$\mathbf{x}_f = [0,1]$
(1)
$w_{1,0} = \Delta G_f^{(b_0)}([0,1])$
$\mathbf{x}_f = [0,0]$
(0)

(a) Full graph $\mathcal{G}_f$.

(b) Example of in-tree on $\mathcal{G}_f$ rooted to state (2).

Fig. 7: Graph $\mathcal{G}_f$ for $B = 2$.

Definition 3: The system of *modified balance equations* for $\mathcal{G}_f$ is:

$$\begin{cases} \max\limits_{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c} \{\nu_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{z}_f)\} = \max\limits_{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c} \{\nu_f(\mathbf{z}_f) - r_f(\mathbf{z}_f, \mathbf{x}_f)\}, \ \forall A \subset \{0,1\}^B \\ \max\limits_{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c} \nu_f(\mathbf{x}_f) = \sigma \end{cases} \tag{20}$$

Property 2. *Given $\{\nu_f(\mathbf{x}_f)\}$ the unique solution of the system for a specific σ, it holds*

$$r_f(\mathbf{x}_f) - \min_{\mathbf{x}'_f} r_f(\mathbf{x}'_f) = \sigma - \nu_f(\mathbf{x}_f). \tag{21}$$

Then, all SS states have the LHS of the equation above equals to 0, so the solution for all SS states are the same (equal σ).

Definition 4: The *potential* of state $\mathbf{x}_f$ is:

$$\phi_f(\mathbf{x}_f) \triangleq G_f(\mathbf{x}_f) - s_f \gamma^\top \mathbf{x}_f. \tag{22}$$

Lemma 2. *The set $\{\phi_f(\mathbf{x}_f), \forall x_f \in \{0,1\}^B\}$ is the solution of the system* (20)*, for a particular value of σ.*

Proof. First, we show that $\max\limits_{\mathbf{x}_f \in A, \mathbf{y}_f \in A^c} \phi_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f)$ is achieved by a pair of parent-child nodes in $\mathcal{G}_f$.

Let $\hat{\mathbf{x}}_f$ and $\hat{\mathbf{z}}_f$ be two nodes in $\mathcal{G}_f$ such that $\hat{\mathbf{x}}_f \in A, \hat{\mathbf{z}}_f \in A^c$, and $|\hat{\mathbf{z}}_f| > |\hat{\mathbf{x}}_f| + 1$. The transition $\hat{\mathbf{x}}_f \to \hat{\mathbf{z}}_f$ has resistance $r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f)$. Now, consider a path from $\hat{\mathbf{x}}_f$ to $\hat{\mathbf{z}}_f$ that traverses nodes with strictly larger weights. By construction, there exists a pair $(\mathbf{x}'_f, \mathbf{y}'_f)$ in this path, such that $\mathbf{x}'_f \in A$ and $\mathbf{y}'_f \in A^c$.

Then,

$$\begin{aligned}
\phi_f(\hat{\mathbf{x}}_f) - r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) &= G_f(\hat{\mathbf{x}}_f) - s_f \gamma^\top \hat{\mathbf{x}}_f - r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) && \text{by def. of } \phi_f(\cdot) \\
&= G_f(\hat{\mathbf{x}}_f) - s_f \gamma^\top \hat{\mathbf{x}}_f - s_f \gamma^\top (\hat{\mathbf{z}}_f - \hat{\mathbf{x}}_f) && \text{by def. of } r_f(\cdot,\cdot) \\
&= G_f(\hat{\mathbf{x}}_f) - s_f \gamma^\top \hat{\mathbf{z}}_f \\
&\leq G_f(\mathbf{x}'_f) - s_f \gamma^\top \hat{\mathbf{z}}_f && \text{by monotonicty of } G_f(\cdot) \\
&= G_f(\mathbf{x}'_f) - s_f \gamma^\top \mathbf{x}'_f - s_f \gamma^\top (\mathbf{y}'_f - \mathbf{x}'_f) - s_f \gamma^\top (\hat{\mathbf{z}}_f - \mathbf{y}'_f) \\
&= \phi_f(\mathbf{x}'_f) - s_f \gamma^\top (\mathbf{y}'_f - \mathbf{x}'_f) - s_f \gamma^\top (\hat{\mathbf{z}}_f - \mathbf{y}'_f) && \text{by def. of } \phi_f(\cdot) \\
&= \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f) - s_f \gamma^\top (\hat{\mathbf{z}}_f - \mathbf{y}'_f) && \text{by def. of } r_f(\cdot,\cdot) \\
&\leq \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f) && \hat{\mathbf{z}}_f - \mathbf{y}'_f \text{ is non-negative}
\end{aligned}$$

Moreover, transitions to ancestors are not valid in the MCs, so we set the reverse edges' resistances to infinite in $\mathcal{G}_f$, i.e., $r_f(\hat{\mathbf{z}}_f, \hat{\mathbf{x}}_f) = +\infty$. As a consequence, the system of modified balance equations can be simplified, considering only parent-child pairs, as follows:

$$\begin{cases} \max\limits_{\substack{\mathbf{x}_f \in A, \mathbf{y}_f \in A^c \\ |\mathbf{y}_f| = |\mathbf{x}_f|+1}} \nu_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f) = \max\limits_{\substack{\mathbf{x}_f \in A, \mathbf{y}_f \in A^c \\ |\mathbf{y}_f| = |\mathbf{x}_f|+1}} \nu_f(\mathbf{y}_f) - r_f(\mathbf{y}_f, \mathbf{x}_f), \ \forall A \subset \{0,1\}^B \\ \max\limits_{\mathbf{x}_f \in A} \nu_f(\mathbf{x}_f) = \sigma \end{cases} \tag{23}$$

Finally, we show that, for every pair parent-child , $\phi_f(\cdot)$ satisfies a pairwise balance equation. Consider a parent-child pair $\mathbf{x}_f, \mathbf{y}_f$ such that $\mathbf{y}_f = \mathbf{x}_f \oplus \mathbf{e}^{(b_0)}$. Then,

$$\begin{aligned} \phi_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f) &= G_f(\mathbf{x}_f) - s_f \gamma^\top \mathbf{x}_f - r_f(\mathbf{x}_f, \mathbf{y}_f) && \text{by def of phi} \\ &= G_f(\mathbf{x}_f) - s_f \gamma^\top \mathbf{x}_f - s_f \gamma^\top (\mathbf{y}_f - \mathbf{x}_f) && \text{by def. of r} \\ &= G_f(\mathbf{x}_f) - s_f \gamma^\top \mathbf{y}_f \\ &= G_f(\mathbf{x}_f) - G_f(\mathbf{y}_f) + G_f(\mathbf{y}_f) - s_f \gamma^\top \mathbf{y}_f \\ &= G_f(\mathbf{y}_f) - \Delta G_f^{(b_0)}(\mathbf{y}_f) - s_f \gamma^\top \mathbf{y}_f && \text{by def. of Delta} \\ &= \phi_f(\mathbf{y}_f) - \Delta G_f^{(b_0)}(\mathbf{y}_f) && \text{by def. of phi} \\ &= \phi_f(\mathbf{y}_f) - r_f(\mathbf{y}_f, \mathbf{x}_f) && \text{by def. of r} \end{aligned}$$

Therefore, $\{\phi_f(\mathbf{x}_f), \forall x_f \in \{0,1\}^B\}$ is the solution of the system (20). □

Property 3. *All SS states have (equal) maximal potential, i.e.,* $\phi_f(\mathbf{x}_f) = \arg\max_{\mathbf{x}'_f} \phi_f(\mathbf{x}'_f)$*, if* $\mathbf{x}_f$ *is SS.*

B. Connection with the Static Optimization Problem

Consider the continuous relaxation Problem 4 as follows:

Problem 5 (ADMin Continuous Relaxation)**.**

$$\underset{\mathbf{Z}}{\text{minimize}} \qquad \sum_{f \in [F]} \sum_{\mathbf{x}_f} Z_f(\mathbf{x}_f) \cdot D_f(\mathbf{x}_f) \tag{24}$$

$$\text{subject to} \qquad \sum_{f \in [F]} \sum_{\mathbf{x}_f} s_f \cdot Z_f(\mathbf{x}_f) \cdot x_f^{(b)} \leq C^{(b)}, \forall b \in [B], \tag{25}$$

$$\sum_{\mathbf{x}_f} Z_f(\mathbf{x}_f) = 1, \forall f \in [F], \tag{26}$$

$$Z_f(\mathbf{x}_f) \geq 0, \forall f \in [F], \forall \mathbf{x}_f \in \{0,1\}^B. \tag{27}$$

In Problem 5, we replace variable $\mathbf{Y}$ with its continuous counterpart $\mathbf{Z} \in \mathbb{R}^{F \times 2^B}$. The continuous relaxation of both capacity and singularity constraints are represented in (25) and (26), respectively. We consider a new set of variables (27) to guarantee that $Z_f(\mathbf{x}_f) \in [0,1], \forall f \in [F], \forall \mathbf{x}_f \in \{0,1\}^B$.

The Lagrangian function of Problem 5 is:

$$\begin{aligned} L(\mathbf{Z}, \boldsymbol{\chi}, \boldsymbol{\zeta}) \triangleq \ & \ell^{\text{OF}}(\mathbf{Z}) + \sum_{b \in [B]} \chi_b \cdot \ell_b^{\text{CC}}(\mathbf{Z}) \\ & + \sum_{f \in [F]} \sum_{\mathbf{x}_f} \chi'_{f,\mathbf{x}_f} \cdot \ell_{f,\mathbf{x}_f}^{\text{PC}}(\mathbf{Z}) + \sum_{f \in [F]} \zeta_f \cdot \ell_f^{\text{SC}}(\mathbf{Z}) \end{aligned} \tag{28}$$

If we replace functions ℓ in (28), we obtain:

$$\begin{aligned} L(\mathbf{Z}, \boldsymbol{\chi}, \boldsymbol{\zeta}) = & \sum_{f \in [F]} \sum_{\mathbf{x}_f} Z_f(\mathbf{x}_f) \cdot D_f(\mathbf{x}_f) \\ & + \sum_{b \in [B]} \chi_b \cdot \left(\sum_{f \in [F]} \sum_{\mathbf{x}_f} s_f \cdot Z_f(\mathbf{x}_f) \cdot x_f^{(b)} - C^{(b)} \right) \\ & - \sum_{f \in [F]} \sum_{\mathbf{x}_f} \chi'_{f,\mathbf{x}_f} \cdot Z_f(\mathbf{x}_f) \\ & + \sum_{f \in [F]} \zeta_f \cdot \left(\sum_{\mathbf{x}_f} Z_f(\mathbf{x}_f) - 1 \right) \end{aligned}$$

Now, we focus on verifying whether the KKT conditions are satisfied for a particular assignment $(\mathbf{Z}^*, \boldsymbol{\chi}^*, \boldsymbol{\zeta}^*)$ defined as follows:

$$\begin{cases} Z_f^*(\mathbf{x}_f) = \pi_{f,0+}(\mathbf{x}_f), & \forall f \in [F], \mathbf{x}_f \in \{0,1\}^B \\ \chi_b^* = \gamma_b, & \forall b \in [B] \\ \chi'^*_{f,\mathbf{x}_f} = \max_{\mathbf{x}'_f} \phi_f(\mathbf{x}'_f) - \phi_f(\mathbf{x}_f), & \forall f \in [F], \mathbf{x}_f \in \{0,1\}^B \\ \zeta_f^* = \max_{\mathbf{x}'_f} \phi_f(\mathbf{x}'_f), & \forall f \in [F], \end{cases}$$

Lagrangian multipliers for the capacity constraints satisfy **Condition 1** because $\boldsymbol{\gamma} \succ 0$. Moreover, if $\mathbf{x}_f$ is an SS state, its potential is maximal, i.e., $\mathbf{x}_f = \arg\max_{\mathbf{x}'_f}\{\phi_f(\mathbf{x}'_f)\}$, then $\chi'^*_{f,\mathbf{x}_f} = 0$ and $\chi'^*_{f,\mathbf{x}_f} > 0$, if $\mathbf{x}_f$ is not SS state. Thus, Lagrangian multipliers for "positivity" constraints are also non-negative and satisfy condition 1.

Condition 2 is satisfied for capacity constraints multipliers χ_b^* because the maximum utility is achieved if the cache space is fully occupied, i.e., $\sum_{f=1}^{F}\sum_{\mathbf{x}_f} s_f \cdot \alpha_f(\mathbf{x}_f) \cdot x_f^{(b)} - C = 0, \forall b$, which is always possible considering the optimization problem's continuous relaxation. Then, $f_b \cdot \chi_b^* = 0, \forall b \in [B]$. For the positivity constraints multipliers: If $\mathbf{x}_f$ is SS state, $\chi'^*_{f,\mathbf{x}_f} = 0$, otherwise, $\alpha_f(\mathbf{x}_f) = 0$, and, therefore, $f'_{i,\mathbf{x}_i} \cdot \chi'^*_{f,\mathbf{x}_f} = 0, \forall i \in [F], \forall \mathbf{x}_i \in \{0,1\}^B$.

Condition 3 is satisfied because the MCs were designed in a way that the expected occupancy (inequality constraints) are used to determine the characteristic time and couple the MCs of different files.

Condition 4 is satisfied because the stationary probabilities must sum 1.

In order for **condition 5** to hold, for each $\alpha_f(\mathbf{x}_f)$,

$$\begin{aligned} &\frac{\partial f_0(\boldsymbol{\alpha})}{\partial \alpha_f(\mathbf{x}_f)} + \sum_{b=1}^{B} \chi_b^* \frac{\partial f_b(\boldsymbol{\alpha})}{\partial \alpha_f(\mathbf{x}_f)} \\ &\quad + \sum_{f=1}^{F} \sum_{\mathbf{x}_f \in \{0,1\}^B} \chi'^*_{f,\mathbf{x}_f} \frac{\partial f_{f,\mathbf{x}_f}(\boldsymbol{\alpha})}{\partial \alpha_f(\mathbf{x}_f)} + \sum_{f=1}^{F} \zeta_f^* \frac{\partial h_f(\boldsymbol{\alpha})}{\partial \alpha_f(\mathbf{x}_f)} \\ &= -G_f(\mathbf{x}_f) + \sum_{b=1}^{B} \gamma_b \cdot s_f \cdot x_f^{(b)} \\ &\quad - \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f) + \phi_f(\mathbf{x}_f) + \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f) \\ &= -\phi_f(\mathbf{x}_f) - \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f) + \phi_f(\mathbf{x}_f) + \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f) \\ &= 0. \end{aligned}$$