HAL
open science

Randomised Geographic Caching and its Applications in Wireless Networks

Anastasios Giovanidis, Bartlomiej Blaszczyszyn

▶ To cite this version:

Anastasios Giovanidis, Bartlomiej Blaszczyszyn. Randomised Geographic Caching and its Applications in Wireless Networks. 2020. hal-02994177v1

HAL Id: hal-02994177
https://hal.science/hal-02994177v1

Preprint submitted on 7 Nov 2020 (v1), last revised 18 Jan 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Chapter 1

Randomised Geographic Caching and its Applications in Wireless Networks

Anastasios Giovanidis[1] *and Bartłomiej Błaszczyszyn*[2]

The randomised (or probabilistic) geographic caching is a proactive content placement strategy that has attracted a lot of attention, because it can simplify a great deal cache-management problems at the wireless edge. It diversifies content placement over caches and applies to scenarios where a request can be possibly served by multiple cache memories. Its simplicity and strength is due to randomisation. It allows one to formulate continuous optimisation problems for content placement over large homogeneous geographic areas. These can be solved to optimality by standard convex methods, and can even provide closed-form solutions for specific cases. This way the algorithmic obstacles from NP-hardness are avoided and optimal solutions can be derived with low computational cost. Randomised caching has a large spectrum of applications in real-world wireless problems, including femto-caching, multi-tier networks, device-to-device communications, mobility, mm-wave, security, UAVs, and more. In this chapter we will formally present the main policy with its applications in various wireless scenarios. We will further introduce some very useful extensions related to unequal file-sizes and content placement with neighbourhood dependence.

1.1 Introduction

The idea to bring multimedia content closer to the user and offload cellular base stations has gained momentum during the research and development phase of 5th generation (5G) cellular networks [1], as well as in the context of Internet-of-Things (IoT). In the seminal paper by Shanmugam et al [2] the authors introduced a novel architecture where so called *helper stations* can be deployed as an additional support to cellular communications. These "helpers" are low complexity stations with weak connection to the backhaul, that are equipped with storage space to cache popular multimedia content. They are seen as small local data-centres which can offload traffic, that otherwise would be served over the internet through the backhaul. This way, congestion can be avoided because each multimedia content needs to be fetched once

[1]Sorbonne University, CNRS-LIP6, Email: anastasios.giovanidis@lip6.fr
[2]Inria-ENS, Paris, France, Email: bartek.blaszczyszyn@ens.fr

and stored at the local helper. Users asking for this content can be served directly by the locally stored copy for so long as this is made available. Furthermore, the density of the helpers can allow for high quality video reception, thus improving the user Quality-of-Experience (QoE). The original idea of installing helpers was gradually extended to include cache-memories on multiple-tiers of the heterogeneous cellular network [3] including the base station. E.g. cache storage space can be installed at the Baseband Unit (BBU) or the BBU pool for Cloud-RAN architectures. Furthermore, the mobile devices [4], [5] can also make use of their own storage space to cache content of interest for the whole network, which can be transmitted to other users via direct device-to-device (D2D) links bypassing the cellular infrastructure. These ideas extend in practice the concept of Content Centric Network architectures [6], to include the wireless-edge.

In all these scenarios, it is very important to decide what content to cache where, and for how long. There are two main types of cache-management policies, the *proactive* [1] and the *online* [7]. The former suggests that the whole memory inventory should be determined in advance based on measurements of local content popularity, and be updated infrequently over long time-windows. The latter favours a frequent update of the cache inventory every time a new request arrives, with the aim to follow closely the evolution of present requests. It includes the Time-To-Live (TTL) and the Least-Recently-Used (LRU) [8], [9] replacement principle.

1.1.1 The FemtoCaching content placement

In the FemtoCaching paper [2] the authors proposed a proactive content placement policy resulting from a binary optimisation problem. They assume all user and station positions are fixed and known. In a dense node deployment with frequency reuse, since one user can be covered by several stations or helpers, the user-station association forms a bipartite graph. Assuming further that a user can be served by any covering station, the content placement variables $x_{i,j} \in \{0,1\}$ are the unknowns, which should decide whether to place content j on the cache of station i, or not. The aim of the placement is to minimise the expected download time for files. In the special case of unit delay [2, eq. 3] the problem simplifies to *maximising sum hit-probability*

$$\begin{array}{lll} \max_{\mathbf{x}} & \sum_{u=1}^{U}\left[1-\sum_{j=1}^{F} a_j \prod_{i\in\mathscr{H}(u)}(1-x_{i,j})\right] & \texttt{[BinCache]} \\ \text{subject to} & \sum_{j=1}^{F} x_{i,j} \leq K, \quad i=1,\ldots,N & \\ & \mathbf{x}\in\{0,1\}^{N\times F} & \end{array} \quad . \tag{1.1}$$

In the above, the stations are indexed by $i = 1,\ldots,N$, the users by $u = 1,\ldots,U$, and the content by $j = 1,\ldots,F$. A user asks for content j with probability a_j. Also, $\mathscr{H}(u)$ is the set of stations covering user u, which can eventually serve the request, if this is available on-cache. The objective function is the expected hit rate summed over all users. To see this, let us focus on user u covered by the station set $\mathscr{H}(u)$. The user can ask for content j with some probability a_j. Then his request will be served by the local helpers (we call this a *hit*), if content j is stored in at least one

cache among the set $\mathscr{H}(u)$. This is written as $1-\prod_{i\in\mathscr{H}(u)}(1-x_{i,j})\in\{0,1\}$. This expression is 0 only if $x_{i,j}=0$ for all covering stations. By taking expectation over the user request probability, we obtain $\sum_{j=1}^{F} a_j\left(1-\prod_{i\in\mathscr{H}(u)}(1-x_{i,j})\right)$ and note that $\sum_{j=1}^{F} a_j = 1$. Finally, summing up over all users U we get the objective.

The constraint set limits the number of content placed in each cache-memory $i=1,\ldots,N$. The authors assume that all files are of equal size - or that these can be broken into chunks of equal size. Then, all cache-memories have a capacity equal to $K\geq 1$ files, K being a natural number. Since the decision variables $(x_{i,1},\ldots,x_{i,F})$ for station i are all binary, their sum should not exceed the capacity K.

We call this problem `BinCache` because it is a binary cache placement problem. Its solution will place a different set of K files on each station, so that higher hit rate is achieved through content diversity. The above is shown to be NP-hard, and the authors propose approximation algorithms for a *sub-optimal* solution with some guaranteed distance from optimality. Their algorithm is based on a continuous relaxation of the unknowns $\tilde{x}_{i,j}\in[0,1]$ and subsequent pipage rounding. It has complexity of the order $O((U+N)^{7/2}F^{7/2})$, which is growing too high for realistic numbers of users, stations and files. For the case when every user is connected to at most 4 helpers (i.e. $card(\mathscr{H}(u))\leq 4$), their approximation ratio is $1-(1-1/4)^4\approx 68\%$ from the optimal. Extensions of the `BinCache` include the joint caching and association problem [10], and the consideration of memory leasing [11], which have similar inherited issues.

1.1.2 The randomised approach

The exploding algorithmic complexity as the number of nodes and the catalog size increases, the sub-optimality of the solution, and the requirement for known fixed positions of a given user set, render `BinCache` impractical. To overcome all these drawbacks, we have suggested in [12] a continuous reformulation of the cache placement problem. The problem and the new policy resulting from its solution are known in the literature as *randomised (or probabilistic) geographic caching*. We refer to the specific problem with cache helpers formulated in [12] and its solution as `RandCache`. The formulation of the simplified problem `RandCache` is based on two principal ideas. (i) The first one is the randomness in user and station positions, assuming homogeneity over the whole area. (ii) The second is the randomness (actually independence) of content placement. These two ideas are of course very much related.

(i) We consider a *typical user* rather than a specific given set of users. Assuming that users take positions homogeneously on the plane with some density per unit area, this representative user can be picked at random among them, and thus has a statistical viewpoint of the network. Let us denote this sampled user by u_0. Since the user was picked at random, the cardinality of stations covering him $card(\mathscr{H}(u_0))$ is a random variable with some distribution. To empirically find this distribution, we calculate the percentage p_0 of users present in the system that are not covered by any station. i.e. those users with $card(\mathscr{H}(u))=0$, then the percentage p_1 of users covered by a single station with $card(\mathscr{H}(u))=1$, and so on. Hence, the

coverage number of the typical user is drawn from the empirical distribution $\{p_m\}$, $m = 0, 1, 2, \ldots$. Note that, this formulation does not need to know the exact coverage set $\mathscr{H}(u)$ per user, just the statistics of the coverage number.

Another way to estimate $\{p_m\}$ is by a measurement campaign as follows: we split the whole planar area with a grid. At each vertex of the grid we measure how many stations cover this point. Then, similarly as above, we can find that approximately p_0 percentage of the points in the grid are not covered by any station, p_1 are covered by exactly one, and so on. The denser the grid, the finer our estimation over the coverage number mass function. The reason we can do so, is that we assume that the requesting user can be selected at random over the plane, and can take uniformly any position. We thus presume an homogeneous distribution of users on the plane.

(ii) Since only the number of covering stations is important and not in detail which station covers which user, it is only natural to consider content placement also probabilistically. A randomised placement ensures that content j needs to be present in a *percentage* of all stations equal to $0 \leq b_j \leq 1$, without specifying which stations in particular. In other words, if we pick a station at random we expect that this station will hold content j in its cache with probability b_j. It is important to note that content should be placed *independently* of the user positions and *independently* of station positions, in order to measure the performance of the typical user.

With the above, the problem `BinCache` becomes continuous and takes the following form, called `RandCache`,

$$\begin{array}{ll} \max_{\mathbf{b}} & \mathbb{E}\left[1 - \sum_{j=1}^{F} a_j (1-b_j)^{card(\mathscr{H})}\right] \quad [\texttt{RandCache}] \\ \text{subject to} & \sum_{j=1}^{F} b_j \leq K, \\ & \mathbf{b} \in [0,1]^F \end{array} . \tag{1.2}$$

Observe here, that the new objective is in expectation, and should be compared with the objective of `BinCache` after dividing the latter by the number of users U to obtain an average expression of hit-probability over the given set of users U. The expectation in `RandCache` is taken over the random variable $card(\mathscr{H})$, i.e. the cardinality of the covering set of base stations, which has probability mass function $\{p_m\}$, for $m = 1, \ldots, M$. Here, M is the maximum number of covering stations and we can allow $M \to \infty$ to include all possible scenarios. Then the objective can be equivalently written as

$$f_{cell}(\mathbf{b}) \quad = \quad 1 - \sum_{j=1}^{F} a_j \sum_{m=1}^{M} p_m (1-b_j)^m . \tag{1.3}$$

Observe that in the randomised version the product $(1-b_j)^m$ replaces the product $\prod_{i \in \mathscr{H}(u)} (1-x_{i,j})$ with $card(\mathscr{H}(u)) = m$ in the binary one. The product $(1-b_j)^m$ is due to *independence* of content placement among stations, i.e. if $i = 1, \ldots, m$ station covers the user $\mathbb{P}\left(\{x_{1,j} = 0\} \cap \ldots \cap \{x_{m,j} = 0\}\right) = \mathbb{P}\left(x_{1,j} = 0\right)^m = (1-b_j)^m$. The probability b_j substitutes every element of the decision vector $(x_{1,j}, \ldots, x_{N,j})$. We could interpret the probability b_j as the percentage of present copies for content

j when the number of stations grows large,

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} x_{i,j} \quad = \quad b_j \quad \in [0,1]\,. \tag{1.4}$$

A direct benefit from the independent randomisation is that the `RandCache` problem has F continuous unknowns compared to the $N \times F$ binary unknowns of the `BinCache` problem, because it treats all stations in a homogeneous way, thus removing the index $i = 1,\ldots,N$. It has, thus, much smaller dimension.

Furthermore, it can be shown that the new objective is concave, and the problem is convex and separable in the variables b_j, so we can find very efficient algorithms to solve it. After having obtained the optimal solution $(b_1^*,\ldots,b_F^*)$, however, it is not obvious how to place content on stations, while respecting these probabilities (frequencies). As we will see, there is a particular way to place batches of K contents on each station, which is very different from the binary policy. We need to emphasise that the `RandCache` is not just a linear relaxation of the `BinCache` problem.

The randomised content placement from [12] found application in the years that followed in many interesting edge-caching scenarios, including:

- *Heterogeneous (multi-tier) networks* with Nakagami fading [13], two-tier [14], and multi-tier placement [3], [15], [16], area spectral efficiency [17], cooperative transmission [18].
- *mm-wave communications* [19], with multiple antennas [20].
- *Device-to-device (D2D) communications* [21], and variations with spatial content correlation [22].
- *Unmanned Aerial Vehicle (UAV) communications* [23], [24].
- *Opportunistic spectrum access* [25].
- *Time-varying content popularity* [26].
- *Effect of retransmissions in throughput* [27].
- *File secrecy and physical-layer security* [28].
- *Joint content caching and recommendations* [29].
- *Content classes of unequal file-size* [30].

A distributed solution for the `RandCache` problem and its variations is proposed in [31] using arguments from game theory and simulated annealing.

In what follows, we will present in detail the randomised policy in Section 1.2. The ingredients of the `RandCache` problem are given in Section 1.3 and its solution steps in Section 1.4. In Section 1.5 we extend the randomised policy, to treat unequal file sizes. Furthermore, we present two additional applications, one for device-to-device networks with mobility in Section 1.6, and one with competition among several content providers for cache space, in Section 1.7. Finally, in Section 1.8 we depart from the assumption of independent random placement among caches to introduce policies of content placement with cache dependence.

1.2 Random Independent Placement Policy

We have shown in the introduction how the randomisation of content placement allowed us to move from the `BinCache` problem to the `RandCache` problem. The random independent placement policy is in fact much more general than the specific problem instance in `RandCache`. Furthermore, this policy has an interesting way to be implemented.

More specifically, in the generic caching problem we assume that there is a set of stations $i = 1, \ldots, N$, and each has a memory of size $K \geq 1$ contents installed. The catalog of possible contents is of size F and contents are indexed by $j = 1, \ldots, F$ and all have *unit size*. The memory inventory of station i contains a subset of the content catalog with a number of elements not larger than K. To keep a general notation that will include many cases, suppose that the decision to place content j in cache i is a binary random variable (r.v.)

$$X_{i,j} \quad = \quad \begin{cases} 1, & \text{with probability } b_{i,j} \in [0,1] \\ 0, & \text{with probability } 1 - b_{i,j} \end{cases} . \tag{1.5}$$

Note that each of these probabilities refers to a specific station i and can differ from one station to the other. These probabilities generally take values within the interval $b_{i,j} \in [0,1]$.

• *Deterministic policies:* To avoid mixed time-sharing policies, a way to proceed is to enforce $b_{i,j} \in \{0,1\}$. This way we can define a *deterministic* policy, where $X_{i,j} = x_{i,j} \in \{0,1\}$ with probability 1. To have a feasible policy, we need to ensure that no more than K items are installed per cache. The set of feasible deterministic placements for our problem with N stations is then defined as

$$\mathscr{F}_{determ} \quad = \quad \left\{ \mathbf{x} \mid \mathbf{x} \in \{0,1\}^{N \times F}, \ \sum_{j=1}^{F} x_{i,j} \leq K, \ i = 1, \ldots, N \right\}. \tag{1.6}$$

The way the set is defined guarantees that at most one item j is placed in each cache i. The above set is the constraint set of `BinCache`. Depending on the objective function of the optimisation problem, its solution will provide an optimal vector $\mathbf{x}^*$ among the feasible ones of size $N \times F$. Once this vector is determined, each cache memory i is filled with the contents given by the solution $x^*_{i,j} = 1$.

• *Randomised homogeneous policies:* A different category of policies results by enforcing the same probabilistic placement for all stations. This would read $b_{i,j} = b_j$ for all stations $i = 1, \ldots, N$ and we refer to it as randomised homogeneous. Each station should place a copy of content j *independently* with probability b_j. We write

$$b_j \quad = \quad \mathbb{P}(X_{i,j} = 1), \ \ i = 1, \ldots, N. \tag{1.7}$$

This would mean that a copy of content j should be present in a b_j percentage of the N stations. Of course, this interpretation makes sense for N large enough.

For a randomised policy to be feasible, each realisation should guarantee that at most K contents are placed in each station, as in the deterministic case. This

constraint reads,

$$\sum_{j=1}^{F} X_{i,j} \quad \leq \quad K, \tag{1.8}$$

where the left hand-side is a random sum. All is left is to find a policy which places content with probability (1.7) in each cache, while satisfying the constraint (1.8). Of course, any such policy will necessarily satisfy in expectation,

$$\mathbb{E}\left[\sum_{j=1}^{F} X_{i,j}\right] = \sum_{j=1}^{F} \mathbb{P}(X_{i,j}=1) \overset{1.7}{=} \sum_{j=1}^{F} b_j \quad \leq \quad K. \tag{1.9}$$

This is not a trivial task. Let us proceed naively and decide to place content j in each cache i by a random coin toss (`Bernoulli` variable) with probability b_j. If we repeat this process independently for all contents $j = 1,\ldots,F$, although we do respect (1.7), however we cannot guarantee constraint (1.8) that there will be less or equal to K different files per cache. To see this, let as use an example with size $K = 2$ memory slots, catalog $F = 4$ contents and a placement vector for the F contents $(0.5, 0.5, 0.5, 0.5)$, such that $\sum_{j=1}^{4} b_j = 2$. If we perform an independent coin toss, the probability to select all 4 contents and overflow the memory space is $(0.5)^4 = 0.0625$ (larger than zero). The independent coin toss (Bernoulli trial) will not guarantee the constraint (1.8). So, we need to propose a different way to place content appropriately.

The idea is as follows: Instead of deciding independently whether to place or not content j in some cache, let us better choose an entire vector of K contents. The K-sized vectors should be drawn independently one from the other, but should respect that the frequency of appearance of each content j is b_j as in (1.7).

Random placement policy

We are given a vector of placement probabilities $\mathbf{b} = (b_1,\ldots,b_F)$ for the F contents of the catalog, satisfying (1.7) and (1.9), and a cache memory of size $K \geq 1$. We split the cache into K continuous memory intervals of unit length and place them one under the other, as shown in Fig. 1.1. The F contents of the catalog are picked one after the other without replacement and in any order. Their values b_j fill in the memory intervals without leaving a gap. When a unit interval is full and cannot contain the entire b_j, the remainder of b_j fills in the next unit memory interval below. The content probabilities b_j are continuously placed one next to the other until all contents have been selected. Due to the constraint (1.9) the process does not exceed the memory space K. To simplify, suppose that the memory is full up to K (although this is not necessary). In order to randomly choose a set of exactly K contents, with a joint distribution that follows (1.7) and (1.9), we pick uniformly at random a number within the interval $[0,1]$ and draw a vertical line intersecting the memory space covered by exactly K distinct contents. The contents are distinct because $b_j \in [0,1]$.

With the above method, the probability of appearance of content j in a memory of size K is exactly equal to b_j.

In a sense, this policy is a *random vector generator of size* $K > 1$, which generalises the discrete random generator (for $K = 1$). Since we have found a way to place content while respecting both (1.7) and (1.8), the feasible set for any randomised homogeneous placement problem becomes,

$$\mathscr{F}_{random} = \left\{ \mathbf{b} \mid \mathbf{b} \in [0,1]^F, \sum_{j=1}^{F} b_j \leq K \right\}. \tag{1.10}$$

The above is just the constraint set for `RandCache`. We can use any objective function that treats all caches homogeneously to derive the optimal feasible solution $\mathbf{b}^*$. Once the optimal probability vector is known, the K-sized memories of all stations can be filled in by randomly drawing vectors using the random placement policy.

In the example of Fig. 1.1 we see that the following batches of $K = 3$ items can be drawn from our method: $\mathscr{C}_1 = \{\text{“1”}, \text{“2”}, \text{“3”}\}$, $\mathscr{C}_2 = \{\text{“1”}, \text{“2”}, \text{“4”}\}$, $\mathscr{C}_3 = \{\text{“1”}, \text{“2”}, \text{“5”}\}$, $\mathscr{C}_4 = \{\text{“1”}, \text{“3”}, \text{“5”}\}$, $\mathscr{C}_5 = \{\text{“2”}, \text{“3”}, \text{“5”}\}$, and $\mathscr{C}_6 = \{\text{“2”}, \text{“3”}, \text{“6”}\}$. What is interesting is that the frequency of each batch depends on the width occupied on the unitary interval. We can observe e.g. that the most frequent batch from the six possible is the $\mathscr{C}_2 = \{\text{“1”}, \text{“2”}, \text{“4”}\}$, because it occupies a wider interval. If we measure the frequency of each batch $(\rho_1, \ldots, \rho_6)$ we can reproduce the values of b_j, $j = 1, \ldots, 6$. To see this, take content “1”. This content is present inside the batches $\mathscr{C}_1, \mathscr{C}_2, \mathscr{C}_3, \mathscr{C}_4$. So, to shift from batch frequencies to item frequencies we just sum-up $\rho_1 + \rho_2 + \rho_3 + \rho_4 = b_1$. In general,

$$b_j = \sum_k \rho_k \mathbf{1}_{\{j \in \mathscr{C}_k\}}, \tag{1.11}$$

where as explained above ρ_k is the frequency of the k-th possible batch, and the indicator function $\mathbf{1}_{\{j \in \mathscr{C}_k\}} = 1$ if content j is inside the k-th batch, otherwise 0.

The random placement policy selects independently, at random a vector of K contents per station, thus it is not a solution tailored for some specific position, contrary to the deterministic policy. It only tries to statistically guarantee some performance of the system, in expectation. This loss in detail is what brings the benefits of continuity, and reduction in the number of unknowns. Furthermore, since the users in real scenarios are mobile, the optimal deterministic solution is bound to change the whole time. Hence, it might be preferable for scenarios with mobility to apply the randomised rather than the deterministic solution. The same holds for scenarios with device-to-device communications, where nodes are mobile and so they change constantly position relative to each other. In both cases, the statistics of coverage number are stable, but the actual instance of the user-station coverage bipartite graph changes constantly. On the other hand, the random vector generator presented above will not work well in cases where the stations are positioned inhomogeneously, i.e. more dense at some areas and less at others. In such scenarios, the random placement policy needs to be appropriately adapted (see also Section 1.8).

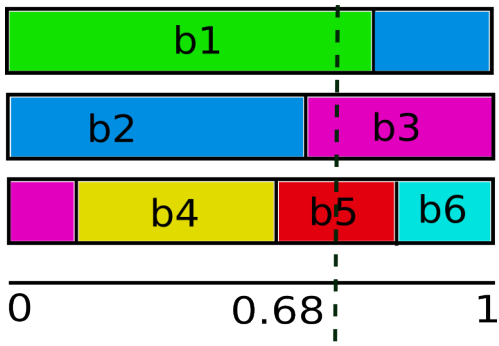

Figure 1.1 *A realisation of the probabilistic placement policy for $F = 6$ contents and $K = 3$ memory slots. We first draw uniformly a random number within the interval $[0,1]$, say we find 0.68. The vertical line at this point intersects with each of the 3 memory slots at a specific content. We conclude from the figure that contents $\{$"1","3","5"$\}$ will be cached.*

1.3 The typical user

We consider a wireless network, where stations are positioned on the two-dimensional (2D) plane. The performance of the network is evaluated at the Cartesian origin $(0,0)$, which we denote as the *typical user* u_o, following our discussion in the introduction. The typical user is characterised by two distributions: one for the content requests, and one for the number of covering stations.

In order for the typical user performance to be representative, the statistics of the network should be unchanged whatever point we choose for the evaluation. For the traffic, this can be achieved if requests network-wide are drawn independently from the same probability distribution. For the coverage number, it depends on how stations are positioned relatively to the users. Assuming that stations are placed following a homogeneous Poisson Point Process (PPP) in 2D with constant density $\lambda_s > 0$ $[stations/unit\ area]$, then such process has the properties of stationarity and isotropy [32] and the typical user approach is valid [33]. We denote the PPP of the stations by Φ_s and a realisation of this process is $\phi_s = \{\chi_i\}$, where χ_i are the deterministic coordinates of the i-th station in this specific realisation. We will evaluate the performance at the typical user over all possible realisations of the process Φ_s.

1.3.1 Coverage number

In cellular networks a user at a random location may be covered by one or more BSs, or may not be covered at all. The so called *coverage number* is a random variable (r.v.) [34], that depends on the features of the communications scheme and the network parameters (such as transmission power). We denote the coverage number by $\mathscr{M}$,

$$p_m := \mathbb{P}[\mathscr{M} = m], \qquad m = 0, 1, \ldots. \tag{1.12}$$

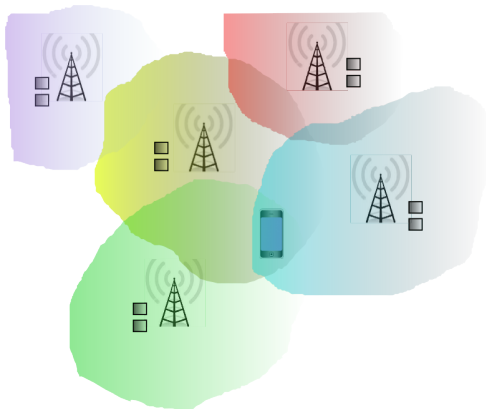

Figure 1.2 Example of a cellular network with caching on Base Stations. A user is covered by three different BSs. Each BS has a cache memory of size two, where content can be temporarily saved. Hence, the user can choose among the content in a total of six memory slots.

The maximum number of covering BSs is $M \in \mathbb{N}^+ \cup \{\infty\}$, where we let the coverage number be unlimited in the general case. This probability mass function p_m can take specific expressions, which depend on the communications scheme evaluated. Obviously, it holds for all cases

$$\sum_{m=0}^{M} p_m = 1. \tag{1.13}$$

In what follows we will present special cases of coverage, depending on the communications scenario. An example of multi-coverage is illustrated in Fig. 1.2.

1.3.1.1 SINR model

Here we consider the downlink of a cellular network and study the signal reception at the typical user. In the most general case all stations operate on the entire available frequency spectrum and the user experiences interference of unrelated stations upon signal reception. The quality of coverage at the origin is described by the SINR_o (from now on SINR). $\text{SINR}(\chi_i)$ is the SINR at the reception, when user u_o is served by BS χ_i and is defined as

$$\text{SINR}(\chi_i) := \frac{S_i/\ell(r_i)}{\sigma_N^2 W + I - S_i/\ell(r_i)}. \tag{1.14}$$

The constant σ_N^2 is the noise power per [Hertz], W is the bandwidth, $I = \sum_{\chi_i \in \Phi} S_i/\ell(r_i)$ is the total received power from the network, $|\chi_i| = r_i$ is the distance of χ_i from o, and $\ell(r) = (Br)^\beta$ is the path-loss function, with constants $B > 0$ and $\beta > 2$. We say that the typical user is covered when $\text{SINR}(\chi_i) > T$, where T is a predefined positive threshold.

The coverage number $\mathscr{M}(T)$ as a function of the threshold indicates how many stations cover the typical user simultaneously and is the r.v.

$$\mathscr{M}(T) \quad = \quad \sum_{\chi_i \in \Phi} \mathbf{1}[\mathrm{SINR}(\chi_i) \geq T]. \tag{1.15}$$

For the coverage of a user who can choose to be served by different BSs in some realization of Φ, we make use of a basic result from [32, Proposition 6.2], see also [35, lemma 5.1.2] a special case of which is given in [36, Lemma 1]. It is shown that if m stations cover a user at SINR level T, then the following inequality holds

$$m \quad < \quad 1 + 1/T. \tag{1.16}$$

For example, when $T \geq 1$ then necessarily $m < 1 + 1/T < 2$ which implies that $m \in \{0,1\}$. Similarly, when $1 > T \geq 1/2$, $m \in \{0,1,2\}$, etc. The authors in [34] (see also [35, (5.26) and (5.35)]), have given explicit expressions for the probability that the typical user is covered by exactly m BSs, given T and a PPP for the BS positions. For general shadowing, they have calculated the probability

$$p_m^{\mathrm{SINR}} \quad := \quad \mathbf{P}[\mathscr{M}(T) = m]. \tag{1.17}$$

The numerical values for p_m^{SINR} can be calculated by the software developed for MATLAB in [37].

1.3.1.2 Boolean model

For the noise-limited case, where the interference is small compared to noise, we can use the Boolean model to calculate the probability of coverage by m BSs. This is a seed-grain model, where the atoms of the PPP are the seeds. Centered on each atom is a 2D sphere $\mathscr{B}(\chi_i, R_b)$ which describes the area of coverage. R_b is a fixed radius that can be expressed by communications quantities. Specifically, if we only consider signal attenuation and no fading, the received signal at the boundary should be larger than the threshold, in order to guarantee coverage, i.e. $(\tilde{B}R_b)^{-\beta} \geq T \Rightarrow R_b = T^{-1/\beta}\tilde{B}^{-1}$.

It is shown in [32, Lemma 3.1] that the number of BSs covering the typical user follows a Poisson distribution with parameter $\nu = \lambda\pi\left(T^{-1/\beta}\tilde{B}^{-1}\right)^2$

$$p_m^{Bool} \quad = \quad \frac{\nu^m}{m!}e^{-\nu}. \tag{1.18}$$

1.3.1.3 Overlaid 2-Network Model

Very often in practice, it occurs that two (or more) networks of the same provider operate in parallel over an area, using different infrastructure (nodes) and orthogonal resources (bandwidth). It is typical, for example, for operators to have one network of base stations for 3G/4G technology and numerous WiFi hotspots within a city. Given that a user may chose between the two to connect to the Internet with his/her cellphone (and assuming for simplicity mathematically independent models of these two networks), the coverage number at the typical user is distributed as the convolution of the coverage probability vectors of the two individual networks

$\mathbf{p}^{(1)} = \left[p_0^{(1)}, \ldots, p_M^{(1)}\right]$ and $\mathbf{p}^{(2)} = \left[p_0^{(2)}, \ldots, p_M^{(2)}\right]$, that is (with $p_m^{(\cdot)} := 0$ for $m < 0$)

$$\mathbf{p}^{2NET} = \mathbf{p}^{(1)} * \mathbf{p}^{(2)} \quad \Rightarrow \quad p_m^{2NET} = \sum_{n=0}^{M} p_n^{(1)} p_{m-n}^{(2)}. \tag{1.19}$$

1.3.1.4 Coverage number from measurements

For general placement of wireless access nodes, the probability vector $\mathbf{p}$ can be estimated by measurements over a whole area. For example, we can imagine that a measurement scenario samples the coverage number at several planar points, and estimates the probability p_0 as the ratio of points not covered at all divided by the total number of samples. Similarly, p_1 is estimated as the ratio of points covered by exactly one station divided by the total number of sample points, and so on.

1.3.2 Content catalog and popularity

Each user – and consequently the typical user at the Cartesian origin – has a request for a specific content (say video file) that he/she wants to receive. In our model, we consider the Independent Reference Model (IRM) for the traffic [38]. This model considers a fixed content library with cardinality F files. The set is denoted by $\mathscr{C} := \{\text{“1”}, \ldots, \text{“}F\text{”}\}$, where each element is an entire file, and indexing follows the file-names $j = 1, \ldots, F$. We consider for now that all content has the same size, normalised to 1. To apply this approach for unequal sized files, we can assume that each file can be divided into chunks of equal size. (The general case of unequal file sizes will be treated in Section 1.5). Furthermore, each content is related to its popularity, which we assume known a priori. We order the content by popularity: “1” is the most popular content, “2” the second most popular, and so on. The popularity follows a distribution $\{a_j\}$. To be consistent with the above ordering, $a_1 \geq a_2 \geq \ldots \geq a_F$. In the IRM model, every request is independent of every other, drawn from the same distribution. Hence, if we focus on the typical user, he/she requests for content j with probability a_j.

Although the type of distribution can be left open, in practice data from measurements indicate that content popularity is Zipf distributed [39], [40]. Zipf distribution considers the order from the most popular file with index $j = 1$ to the least popular with index $j = F$. The probability that a user requests content j is equal to

$$a_j \quad = \quad C^{-1} \cdot j^{-\gamma}. \tag{1.20}$$

The Zipf exponent γ, is chosen often within the range $\gamma \in (0, 2]$, to limit the difference between the two most popular files, i.e. $a_1/a_2 \leq 4$, but it can vary depending on the s cenario. The normalisation factor C is found by summing the mass to unity

$$\sum_{j=1}^{F} a_j \quad = \quad 1. \tag{1.21}$$

A value γ close to zero results in a uniform popularity distribution, whereas larger values of γ produce distributions with increasingly lighter tails. So in Zipf, γ controls the skewness of the popularity.

1.4 Optimal Randomised Caching in Cellular Networks

In this section, we will present the original problem of caching in cellular networks with multi-coverage and present the optimal (independently) randomised policy published in [12]. The solution will be described in more detail here. As mentioned in the introduction, when searching for the optimal random placement policy, the problem obtains the `RandCache` formulation. The objective (1.3) aims to maximise the probability that the typical user will find the content he/she is asking for in one of the BSs he/she is covered from. This is 1-minus the probability that the user does not find its content anywhere among the covering stations. More precisely, this happens when the user is covered by $m = 0$ stations (i.e. no coverage), or by some number $m > 0$, but the content has not been saved in the cache memory space of any of these. We repeat at this point the optimisation problem [`RandCache`] (1.2),

$$\begin{array}{lll} \max_{\mathbf{b}} & f_{cell}(\mathbf{b}) := 1 - \sum_{j=1}^{F} a_j \sum_{m=0}^{\infty} p_m (1-b_j)^m & [\texttt{RandCache}] \\ \text{s.t} & \mathbf{b} \in \mathscr{F}_{random} & \end{array} \tag{1.22}$$

where the feasible set $\mathscr{F}_{random}$ is defined in (1.10). We can control the objective hit-probability, by varying the content availability b_j, $j = 1, \ldots, F$ within the feasibility region. The size of the cache memory K, plays an important role in the constraints. We further note that, in the case that for some content j the optimal placement is $b_j = 1$, i.e. the content j is cached almost surely in all stations, then $\sum_{m=0}^{\infty} p_m (1 - b_j)^m \mid_{b_j=1} = p_0 (1-b_j)^0 \mid_{b_j=1} = p_0$, i.e. the non-coverage probability.

Remark 1: *For the objective in (1.22) (same as (1.3)) we can make the following observation. Let $z_j := 1 - b_j$. Then we have*

$$\psi(z_j) := \mathbb{E}\left[z_j^{\mathscr{M}}\right] = \sum_{m=0}^{\infty} p_m z_j^m. \tag{1.23}$$

The expression is the probability-generating function (or $\mathscr{Z}$-transform when using z^{-1} instead of z) of the random variable $\mathscr{M}$ defined in (1.12) as the number of stations covering the typical user. By replacing this in (1.3) we get

$$f_{cell}(\mathbf{b}) \quad := \quad 1 - \sum_{j=1}^{J} a_j \psi(1-b_j). \tag{1.24}$$

Before we move on to the general solution of [`RandCache`], let us first formulate and solve a toy version of the problem, with $F = 2$ different types of content, $K = 1$ memory slot per BS and $M = 2$, meaning that the typical user can only be in three coverage states, i.e. 0-coverage, 1- and 2-coverage, with $m \in \{0, 1, 2\}$. Its solution will provide intuition on how to solve the most general one.

$$\begin{array}{lll} \max_{(b_1,b_2)} & \sum_{j=1}^{2} a_j \left(1 - p_0 - p_1(1-b_j)^1 - p_2(1-b_j)^2\right) & [\texttt{RandK1F2}] \\ \textbf{s.t.} & b_1 + b_2 \leq 1 & \\ & b_1, b_2 \in [0,1] & \end{array} .$$

1.4.1 Solution for [`RandK1F2`]

The toy problem can be further simplified by making use of the equalities $p_0 + p_1 + p_2 = 1$ and $a_1 + a_2 = 1$. We also observe that the objective function is *monotone increasing* in the unknowns b_1 and b_2 for their range of values in the feasible set. As a consequence, the optimal solution should satisfy the constraint with equality $b_1 + b_2 = 1$. Using these three equalities, the problem can be expressed only with the parameter set (a_1, p_1, p_2) and has a single unknown b_1

$$\begin{array}{ll} \max & -p_2 b_1^2 + b_1 (2a_1 (p_1 + p_2) - p_1) + (1 - a_1)(p_1 + p_2) \quad \texttt{[RandK1F2-r]} \\ \textbf{s.t.} & b_1 \in [0,1] \end{array}.$$

We denote the objective function of [`RandK1F2-r`] by $f_{toy}(b_1)$. Now, the only constraint is that $b_1 \in [0,1]$. Let us take the first derivative of the objective function,

$$\frac{df_{toy}}{db_1} = -2p_2 b_1 + 2a_1 (p_1 + p_2) - p_1.$$

When the derivative is positive the function is increasing, otherwise decreasing.

$$\frac{df_{toy}}{db_1} \geq 0 \bigcap b_1 \in [0,1] \quad \Rightarrow \quad 0 \leq b_1 \leq \min\left\{1, \frac{2a_1 (p_1 + p_2) - p_1}{2p_2}\right\},$$

where in the above we limit the b_1 inside the interval $[0,1]$. We observe that the objective function is increasing in b_1 until some upper value b_1^*. If this upper value is larger than 1, then the objective is increasing in the entire interval $[0,1]$. Otherwise, it is increasing up to b_1^* and then decreasing in the interval $[b_1^*, 1]$. Then the solution of the problem is simply equal to b_1^*. This is because, as long as b_1 moves from 0 and larger, the objective is also increasing (positive derivative). This is true until b_1^* with objective $f_{toy}(b_1^*)$, after which the objective value decreases. In fact, it can be shown from the second derivative that the function is concave

$$\frac{d^2 f_{toy}}{db_1^2} = -2p_2 \leq 0, \tag{1.25}$$

so there is a unique maximum at b_1^*. To sum up,

Proposition 1: *The solution for the* [`RandK1F2-r`] *problem is given by the placement probability for content type* "1" *equal to*

$$b_1^* = \begin{cases} \frac{2a_1(p_1+p_2)-p_1}{2p_2} & \text{, if } 0.5 \leq a_1 < 1 - \frac{p_1}{2(p_1+p_2)}. \\ 1 & \text{, otherwise} \end{cases} \tag{1.26}$$

The condition for $b_1^* < 1$ is that the popularity of file "1" is within the interval $0.5 \leq a_1 < 1 - p_1/(2p_1 + 2p_2) < 1$. This results from the feasibility constraint $0 \leq b_1^* < 1$, by replacing $b_1^* = \frac{2a_1(p_1+p_2)-p_1}{2p_2}$. The right hand-side is then found directly. The left hand-side inequality gives $a_1 \geq p_1/(2p_1 + 2p_2)$, which is always true because $p_1/(2p_1 + 2p_2) \geq 0.5$ and the files are ordered in popularity, so for $F = 2$ necessarily $a_1 \geq 0.5$, being the most popular one.

Interestingly, as (1.26) shows, the optimal solution b_1^* is just an increasing affine function of the popularity a_1. The more popular the file "1" the more often this file

is placed. The optimal value ranges from $b_1^* = 0.5$ when $a_1 = 0.5$, to $b_1^* = 1$ when $a_1 = 1 - \frac{p_1}{2(p_1+p_2)}$. Obviously $b_2^* = 1 - b_1^*$.

To evaluate the solution, we first plot of the objective function $f_{toy}(b_1)$ over b_1, and for various values of p_1 and p_2. To facilitate comparison, we keep $a_1 = 0.6$ for the popularity of the first content and the sum $p_1 + p_2 = 0.95$, so that $p_0 = 0.05$. For each pair (p_1, p_2) we find the maximum b_1^*. We denote the maximal value of the objective function by $f_{toy}^* = f_{toy}(b_1^*)$. Our findings are summarised in the table below:

p_1	p_2	b_1^*	f_{toy}^*
0.25	0.7	0.6357	0.6629
0.55	0.4	0.7375	0.5976
0.75	0.2	0.9750	0.5701
0.85	0.1	1	0.5700

Fig. 1.3 that follows illustrates the related plots. To explain better we note that

- All curves are concave.
- For $b_1 = 0$: $f_{toy}(0) = (1 - a_1)(p_1 + p_2) = a_2(1 - p_0)$. In our case $(1 - 0.6) * 0.95 = 0.38$ for all curves.
- For $b_1 = 1$: $f_{toy}(1) = a_1(p_1 + p_2)$. In our case $0.6 * 0.95 = 0.57$ for all curves.
- For $b_1 = b_1^*$ there are two cases:
  - If $b_1^* = 1$ then $f_{toy}(b_1^*) = a_1(p_1 + p_2)$.
  - If $b_1^* = \frac{2a_1(p_1+p_2)-p_1}{2p_2} < 1$, then the maximal value is equal to

$$f_{toy}(b_1^*) \quad = \quad f_2(0) + (b_1^*)^2 p_2. \tag{1.27}$$

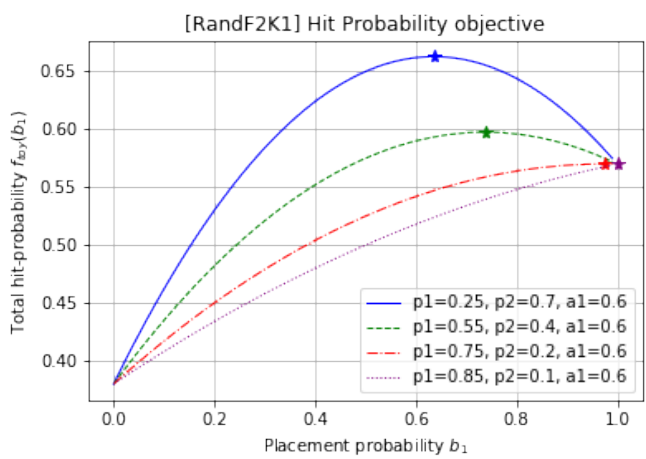

Figure 1.3 Four curves of the hit probability (objective of [`RandK1F2-r`]), for different values of p_1 and p_2, but for constant sum $p_1 + p_2 = 0.95$ and popularity $a_1 = 0.6$. The star marks the maximum.

Even in this simple toy example, the policy of placing the most popular content "1" everywhere ($b_1 = 1$), is not necessarily optimal. When the probability of coverage by 2 BSs is considerable, a portion of the slots should be occupied by the second most popular content "2". This depends also on the content popularity ratio a_2/a_1. To understand this, we plot in in Fig. 1.4 (left) the optimal caching policy as a function of p_1/p_2. We see that on the left part of the x-axis, where the probability of coverage by 2 stations is much larger than by a single one, the placement frequency is split among file "1" and "2", whereas at the right part of the x-axis, where most users are covered by a single station only file "1" is placed. In Fig. 1.4 (right) the total hit is plotted, where it is clearly seen at the right part of the x-axis, that placing both items with the appropriate frequency has important performance improvements. The hit probability with $b_1 = 1$ everywhere at the right part of the x-axis, is simply equal to $a_1(1-p_0)$, i.e. the probability to be covered by at least 1 station and ask for content "1".

Conclusions: This toy example shows that the solution can be found from a convex program with equality constraint. The optimal policy is characterised by diversity of content placement, in rich multi-coverage environments. Always caching the most popular contents is suboptimal in general.

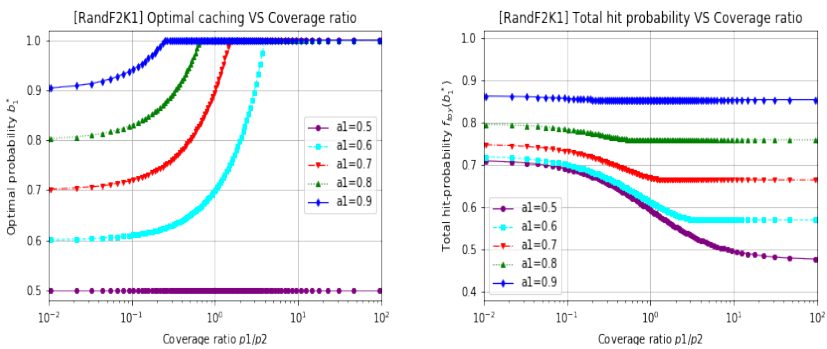

Figure 1.4 Variation of (left) the optimal caching policy b_1^*, and (right) the optimal hit-probability objective, with respect to the coverage ratio p_1/p_2. The evaluation takes five different values of a_1 into consideration.

1.4.2 Problem Solution [`RandCache`]

We move now to the solution of the general problem. The analysis and optimisation is very much facilitated by identifying certain properties of the problem at hand.

Lemma 1: *The objective function of [`RandCache`] in (1.22) has the following two properties:*

- ***P.1:*** *It is separable with respect to the optimisation variables $b_1, \ldots, b_F$ and can be re-written as*

$$f_{cell}(\mathbf{b}) = \sum_{j=1}^{J} a_j \left(1 - \sum_{m=0}^{\infty} p_m (1-b_j)^m\right) = \sum_{j=1}^{F} a_j g(b_j). \tag{1.28}$$

- ***P.2:*** *It is an increasing and concave function of b_j, $\forall j$. Consequently it is a concave function of the vector $(b_1, \ldots, b_F)$.*

Proof. Property 1 results by rewriting the objective function after replacing $1 = \sum_{j=1}^{J} a_j$. The proof of Property 2 becomes trivial due to the separability property. We can see that $\forall j$:

$$\begin{aligned} \frac{dg}{db_j} &= \sum_{m=1}^{\infty} p_m m (1-b_j)^{m-1} \geq 0, \\ \frac{d^2 g}{db_j^2} &= -\sum_{m=2}^{\infty} p_m m(m-1)(1-b_j)^{m-2} \leq 0. \end{aligned}$$

Hence the objective function is increasing (non-decreasing) and concave on b_j, $\forall j$. Since by **P.1** it is also separable, the Hessian is a diagonal matrix, with negative entries on the diagonal equal to $a_j \frac{d^2 g}{db_j^2}$ and hence negative semidefinite. We conclude that the objective function is a concave function of the vector $(b_1, \ldots, b_F)$. □

Since the objective function is concave by **P.2** and the constraint set $\mathscr{F}_{random}$ is linear, the optimisation problem can be solved as a *convex program*. We will make use of the Lagrangian relaxation and the subgradient method (see [41] and [42]).

1.4.2.1 Lagrangian Relaxation

We first relate the dual variable $\mu \geq 0$ to the sum constraint inequality (1.9). This is called a Lagrange multiplier. We write down the Lagrangian function of the optimisation problem

$$L(b_1, \ldots, b_F, \mu) = \sum_{j=1}^{F} a_j \left(1 - \sum_{m=0}^{\infty} p_m (1-b_j)^m\right) + \mu \left(K - \sum_{j=1}^{F} b_j\right). \tag{1.29}$$

The remaining constraint set is

$$\mathscr{F}_2 := \left\{ b_j \in [0,1],\ \forall j = 1, \ldots, J \right\}. \tag{1.30}$$

We can systematically find the optimal primal and dual variables by solving a min-max problem. Additionally, in our case where we deal with a convex program, the optimal value of the min-max problem is equal to the optimal value of the original problem [`RandCache`] with objective function f_{cell}

$$\begin{aligned} f_{cell}^* := \max_{\mathscr{F}_{random}} f_{cell}(b_1, \ldots, b_F) &= \min_{\mu \geq 0} \max_{\mathscr{F}_2} L(b_1, \ldots, b_F, \mu) \\ &= f_{cell}(b_1^*, \ldots, b_F^*). \end{aligned} \tag{1.31}$$

We then say that the duality gap between the original [`RandCache`] problem and the min-max problem is zero. We should further mention that the optimal primal and dual variables b_j^*, $\forall j$ and μ^* are related by the *Complementary Slackness Condition* (CSC), which basically forces the dual variable to be zero when the constraint is active.

$$\mu^* \left(K - \sum_{j=1}^{J} b_j^* \right) = 0 \quad \Rightarrow \quad \begin{cases} \mu^* = 0, & \text{if } K - \sum_{j=1}^{J} b_j^* > 0. \\ \mu^* \geq 0, & \text{if } K - \sum_{j=1}^{J} b_j^* = 0. \end{cases} \tag{1.32}$$

The following Lemma characterizes the solution:

Lemma 2: *At the optimal solution, the sum constraint inequality (1.9) is inactive, i.e. the optimal solution satisfies*

$$b_1^* + \ldots + b_F^* \quad = \quad K. \tag{1.33}$$

Proof. Suppose that the inequality is inactive, i.e. strictly $< K$ for the optimal solution. But then, for any $\ell \leq F$, we can increase $b_\ell^* \to b_\ell^* + \varepsilon$, so that the constraint is satisfied with equality. Substituting in the objective function $b_\ell^* + \varepsilon$ instead of b_ℓ^* the value of the function will increase, because f_{cell} is increasing over b_ℓ (by property **P.2**). Hence the primal optimal solution cannot leave the constraint inactive and (1.33) is true. □

1.4.2.2 Relaxed Primary Problem Decomposition

Taking into account Lemma 2 and the CSC in (1.32), we first consider a fixed dual price $\mu \geq 0$ and solve the *relaxed primary problem* of (1.31), which we rewrite as

$$q(\mu) \quad = \quad \max_{\mathscr{F}_2} L(b_1, \ldots, b_F, \mu). \tag{1.34}$$

From Lemma 1 we can see that the above problem is separable in the variables b_j, i.e. given μ we should solve F separate problems, one for each content

$$\begin{array}{ll} \max & -\mu b_j + a_j (1 - p_0 - \sum_{m=1}^{\infty} p_m (1 - b_j)^m) \quad \text{[subRand-j]} \\ \textbf{s.t.} & 0 \leq b_j \leq 1 \end{array}.$$

Lemma 3: *Given fixed $\mu \geq 0$, the solution of [`subRand-j`] is*

$$b_j(\mu) \quad = \quad \begin{cases} 1, & \text{if } a_j p_1 > \mu \\ \omega(\mu), & \text{if } a_j p_1 \leq \mu \leq a_j \mathbb{E}[\mathscr{M}] \\ 0, & \text{if } a_j \mathbb{E}[\mathscr{M}] < \mu \end{cases}, \tag{1.35}$$

where $\mathbb{E}[\mathscr{M}] = \sum_{m=1}^{\infty} m p_m$ and $\omega(\mu)$ is the solution over b_j of the equation

$$a_j \sum_{m=1}^{\infty} p_m m (1 - b_j)^{m-1} = \mu. \tag{1.36}$$

Proof. We take the first derivative of the objective function of the [`subRand-j`]

$$-\mu + \frac{dg}{db_j} \quad = \quad -\mu + a_j \sum_{m=1}^{\infty} m p_m (1 - b_j)^{m-1}. \tag{1.37}$$

We distinguish between three cases:

- if $-\mu + \frac{dg}{db_j} > 0$, in the entire domain of b_j, the objective function is increasing and the optimal solution is $b_j = 1$. Since from (**P.2**) it holds $d^2g/db_j^2 \leq 0$, the function $-\mu + \frac{dg}{db_j}$ is decreasing in b_j, taking its minimum value for $b_j = 1$. Hence, it suffices to verify that $-\mu + \frac{dg}{db_j}\,|_{b_j=1} > 0$, which from the expression in (1.37) gives the condition $a_j p_1 > \mu$.
- if $-\mu + \frac{dg}{db_j} < 0$, in the entire domain of b_j, the objective function is decreasing and the optimal solution is $b_j = 0$. With the same argument about concavity, the maximum value of $-\mu + \frac{dg}{db_j}$ is obtained for $b_j = 0$. Hence, it suffices to verify $-\mu + \frac{dg}{db_j}\,|_{b_j=0} < 0$, which from (1.37) gives the condition $\mu > a_j \sum_{m=1}^{\infty} m p_m$.
- In the intermediate case, there exists a point $b_j = \omega \in (0,1)$, such that $-\mu + \frac{dg}{db_j} = 0$. The exact value ω is found by solving (1.36).

□

Having written the primal variables as functions of the dual $b_j(\mu)$ in (1.35) we only need to solve the dual problem over the dual variable μ, see (1.31). Using the notation in (1.34), the dual problem is

$$\min_{\mu \geq 0} \; q(\mu). \tag{1.38}$$

1.4.2.3 Subgradient Method for the Dual

One of the most standard ways to solve (1.38) is by the *subgradient method*, which can find the optimal μ^* in a finite number of steps, even in the case where the function $q(\mu)$ is non-differentiable. The method involves a sequence of updates for the value of $\mu^{(k)}$, $k = 1, \ldots$. For each $\mu^{(k)}$ the relaxed primary problem is solved and the solution over $b_j\left(\mu^{(k)}\right)$ is used for the next update that gives $\mu^{(k+1)}$. The method converges when $\left|\mu^{(k+1)} - \mu^{(k)}\right| < \varepsilon$ or ideally $\left|\mu^{(k+1)} - \mu^{(k)}\right|$=0. The update steps are described by the equation

$$\mu^{(k+1)} \quad = \quad \left[\mu^{(k)} + s^{(k)}\left(K - \sum_{j=1}^{F} b_j\left(\mu^{(k)}\right)\right)\right]^+, \; k = 1, 2, \ldots. \tag{1.39}$$

The term in the parenthesis is the *subgradient* of our problem (practically it is the subgradient of the Lagrangian in (1.29) taking $\frac{dL}{d\mu}$), whereas the operation $[\ldots]^+$ sets to zero the case of negative value inside the brackets, to guarantee $\mu^{(k)} \geq 0$. Finally, $s^{(k)}$ is the step size of the iteration, which is determined a priori. For decreasing $s^{(k)}$, and more precisely for a sequence that is *non-negative, non-summable and diminishing*, i.e $s^{(k)} \geq 0$, $\lim_{k\to\infty} s^{(k)} = 0$ and $\sum_{k=1}^{\infty} s^{(k)} = \infty$ [43], the method is guaranteed to converge to the optimal solution, i.e. $q(\mu^{(k)}) \overset{k\to\infty}{\to} q(\mu^*) = f^*_{cell}$. Following the analysis in [43, par. 3.3] the optimal choice in the case of our problem is $s^{(k)} = K/\sqrt{k}$.

1.4.2.4 A tailored method

Our problem [`RandCache`] has a special structure that allows for a tailored method for the dual problem. This may be simpler to implement and faster to converge than the subgradient one. Specifically, from Lemma 2 we know that the optimal solution satisfies the sum constraint with equality. We will prove the following Theorem, that allows to build our own special algorithm

Theorem 1: *As μ decreases from $\mu^{high} = a_1 \mathbb{E}[\mathscr{M}]$ (or increases from $\mu^{low} = a_K p_1$) the sum $\sum_{j=1}^{F} b_j(\mu)$ crosses the horizontal line K at a* ***unique*** *μ^*, such that*

$$b_1(\mu^*) + \ldots + b_F(\mu^*) \quad = \quad K, \tag{1.40}$$

where $b_j(\mu)$ is given in (1.35). The value μ^* is the optimal one for the dual variable. Furthermore, the optimal primal variables are found by replacing $\mu = \mu^*$ in Lemma 3, i.e. $b_j^* = b_j(\mu^*)$, $\forall j$.

We will prove this using Lemma 2 and 3, to first derive Lemma 4 and Lemma 5.

Lemma 4: *The solution $b_j(\mu)$, $j = 1, \ldots, F$ of the relaxed primary problem, given in (1.35), is non-increasing in μ.*

Proof. To see this, let μ_1 and μ_2 be two distinct values of the dual variable. Without loss of generality, assume $\mu_1 > \mu_2$. We want to show that

$$\mu_1 > \mu_2 \quad \Rightarrow \quad b_j(\mu_1) \leq b_j(\mu_2), \; \forall j. \tag{1.41}$$

We write the expression of $b_j(\mu)$ as a combination of all cases in (1.35) using indicator functions

$$b_j(\mu) \quad = \quad \mathbf{1}_{\{a_j p_1 > \mu\}} + \omega(\mu)\mathbf{1}_{\{a_j p_1 \leq \mu \leq a_j \mathbb{E}[\mathscr{M}]\}} + 1 - \mathbf{1}_{\{a_j \mathbb{E}[\mathscr{M}] < \mu\}}. \tag{1.42}$$

Then for $\mu_1 > \mu_2$ we have

$$\begin{aligned} \mathbf{1}_{\{a_j p_1 > \mu_1\}} &\leq \mathbf{1}_{\{a_j p_1 > \mu_2\}} \\ 1 - \mathbf{1}_{\{a_j \mathbb{E}[\mathscr{M}] < \mu_1\}} &\leq 1 - \mathbf{1}_{\{a_j \mathbb{E}[\mathscr{M}] < \mu_2\}}. \end{aligned}$$

For the remaining term, we observe from (1.36) that $\omega(a_j p_1) = 1$, whereas $\omega(a_j \mathbb{E}[\mathscr{M}]) = 0$. Also, the $\omega(\mu)$ is decreasing in μ. To see this,

$$\begin{aligned} \mu_1 > \mu_2 \quad &\Rightarrow \quad a_j \sum_{m=1}^{\infty} p_m m (1 - b_j^{(1)})^{m-1} > a_j \sum_{m=1}^{\infty} p_m m (1 - b_j^{(2)})^{m-1} \\ &\Rightarrow \quad \sum_{m=1}^{\infty} p_m m \left[(1 - b_j^{(1)})^{m-1} - (1 - b_j^{(2)})^{m-1} \right] > 0. \end{aligned} \tag{1.43}$$

The sign of $(1 - b_j^{(1)})^{m-1} - (1 - b_j^{(2)})^{m-1}$ is the same for any $m > 1$, so it should definitely hold $(1 - b_j^{(1)})^{m-1} - (1 - b_j^{(2)})^{m-1} > 0$, for all m, hence also for $m = 2$. This gives the relation $b_j^{(2)} > b_j^{(1)}$. This, combined with the other two observations and the expression in (1.42) proves that (1.41). The inequality is strict $\forall j$ only when $a_1 p_1 < \mu_2 < \mu_1 < a_F \mathbb{E}[\mathscr{M}]$ (worst case). □

Lemma 5: *For $K < F$, the sum function $\sum_{j=1}^{F} b_j(\mu)$ is strictly decreasing for $\mu \in [a_K p_1, a_1 \mathbb{E}[\mathscr{M}]]$. It is equal to 0 when $\mu > a_1 \mathbb{E}[\mathscr{M}]$ and it is larger than K when $\mu < a_K p_1$.*

Proof. To see this, we use the previous Lemma and the expression in (1.42) for the $b_j(\mu)$. The sum of the b_js is necessarily non-increasing. To show that it is strictly decreasing in the claimed interval, we observe that the sum will decrease as long as at least one b_j decreases. The domain of μ where no b_j changes is either for $\mu < \min_j a_j p_1 = a_F p_1$ and all $b_j = 1$ or for $\mu > \max_j a_j \mathbb{E}[\mathscr{M}] = a_1 \mathbb{E}[\mathscr{M}]$ and all $b_j = 0$. The complement of this domain is the claimed interval.

But since there are only $K < F$ slots available in the memory, there cannot be more than K most popular contents with $b_j = 1$, hence we replace the lower limit of the interval by $a_K p_1$.

□

Since the sum is monotone, then the solution μ^* is unique and we have also identified the interval where the solution lies, i.e. $\mu^* \in [a_K p_1, a_1 \mathbb{E}[\mathscr{M}]]$. Using this, we can propose an algorithmic solution.

Algorithm: The proposed numerical algorithm to find the solution $\mathbf{b}^*$ with some given precision is based on the **bisection method**.

- Step $\ell = 0$: Initialise the interval that contains the solution,

$$[\mu^{(0,low)}, \mu^{(0,high)}] \quad = \quad [\, a_K p_1, a_1 \mathbb{E}[\mathscr{M}] \,].$$

  Also, choose a desired precision $\varepsilon > 0$.
- Iterate for $\ell = 0, 1, \ldots$:
  - Find the mid-point of the interval
  $$\mu^{(\ell+1)} \quad = \quad \mu^{(\ell,low)} + \frac{1}{2}\left(\mu^{(\ell,high)} - \mu^{(\ell,low)}\right)$$
  - Evaluate the sum
  $$S^{(\ell+1)} \quad = \quad \sum_{j=1}^{F} b_j\left(\mu^{(\ell+1)}\right)$$
  - If $S^{(\ell+1)} < K$ then $\mu^{(\ell+1,high)} = \mu^{(\ell+1)}$. Else $\mu^{(\ell+1,low)} = \mu^{(\ell+1)}$.
- Repeat the iteration until

$$\left\| \mu^{(\ell)} - \mu^{(\ell+1)} \right\| \quad < \quad \varepsilon.$$

The method converges *linearly* and every iteration halves the distance of $\mu^{(\ell)}$ from the optimal value μ^*. In fact, one needs at most $\log_2(a_1 \mathbb{E}[\mathscr{M}] - a_K p_1) - \log_2(\varepsilon)$ iterations, to achieve ε-precision.

A difficulty encountered when applying the algorithm is to find the values for $b_j\left(\mu^{(\ell+1)}\right)$. These satisfy the polynomial equalities of the form (1.36), which do not have a closed form solution when M is large. We can also solve these equalities over the individual b_j's by use of the *bisection method*.

1.4.2.5 Verification for the [RandK1F2]

We can now verify the solution of [RandK1F2] in (1.4.1) using Theorem 1. Because $m \in \{0,1,2\}$, the $\omega(\mu)$ obtains an explicit expression and (1.35) now becomes

$$b_1(\mu) = \begin{cases} 1, & \text{if } a_1 p_1 > \mu \\ 1 + \frac{a_1 p_1 - \mu}{2 a_1 p_2}, & \text{if } a_1 p_1 \leq \mu \leq a_1(p_1 + 2p_2) \\ 0, & \text{if } a_1(p_1 + 2p_2) < \mu \end{cases}, \tag{1.44}$$

and similarly for $b_2(\mu)$.

Since $K = 1$, the case $b_1^* = 1$, $b_2^* = 0$ happens only when $a_1 p_1 > \mu > a_2(p_1 + 2p_2)$. The last inequality, with $a_1 + a_2 = 1$ and solving for a_1 is rewritten as $a_1 > 1 - \frac{p_1}{2(p_1+p_2)}$. The case where $b_1^* = 0$, $b_2^* = 1$ holds when $a_2 p_1 > \mu > a_1(p_1 + 2p_2)$ and the inequality is rewritten as $a_1 < \frac{p_1}{2(p_1+p_2)}$.

In the case where $b_1^* \in (0,1)$ and $b_2^* \in (0,1)$ we solve the equation $b_1(\mu) + b_2(\mu) = 1$ to find μ^*, i.e.

$$\begin{aligned} 1 &= 1 + \frac{a_1 p_1 - \mu^*}{2 a_1 p_2} + 1 + \frac{a_2 p_1 - \mu^*}{2 a_2 p_2} \Rightarrow \\ \mu^* &= 2 a_1 (1 - a_1)(p_1 + p_2), \quad \textit{with } a_2 = 1 - a_1. \end{aligned} \tag{1.45}$$

We see from the middle condition in (1.44) that

$$\begin{aligned} a_1 p_1 \leq \mu^* = 2 a_1 (1 - a_1)(p_1 + p_2) < a_1(p_1 + 2p_2) \quad &\Rightarrow \\ 1 - \frac{p_1}{2(p_1 + p_2)} \geq a_1 > \frac{p_1}{2(p_1 + p_2)} \quad &. \end{aligned} \tag{1.46}$$

Given the value of μ^* in (1.45) the optimal primal variables are $b_1^* = b_1(2 a_1 a_2 (p_1 + p_2)) = 1 + \frac{a_1 p_1 - 2 a_1 a_2 (p_1 + p_2)}{2 a_1 p_2} = \ldots = \frac{2 a_1 (p_1 + p_2) - p_1}{2 p_2}$. Hence, we result in the same solution as before (see (1.26)).

1.4.2.6 Evaluation of [RandCache]

We can now evaluate the user hit probability when the optimal randomised placement policy is implemented. In this specific problem instance, the coverage number probability in Section (1.3.1.2), is Poisson distributed (1.18) with $M = \infty$, and parameter $\nu = \pi(T^{-1/\beta})^2$. With fading exponent $\beta = 4$, we choose a range of Threshold values $T \in \left[10^{-2}, 10^3\right]$, $F = 25$ files in the catalog and a cache memory equal to $K = 5$. We plot in Fig. 1.5 the optimal hit-probability by solving the [RandCache] problem. Note that the popularity is Zipf, with three possible exponents $\gamma \in \{0.56,\ 0.9,\ 1.6\}$.

We observe that for all popularity distributions, the total hit probability (objective function) tends to 1 when the threshold gets smaller. A smaller threshold implies a larger ν and hence a larger expected value $\mathbb{E}[\mathscr{M}] = \nu$. So, the optimal cache placement can take full advantage of a rich multi-coverage environment and offer diversity of files to the user, so that the hit-probability reaches 1. Of course, the hit probability tends to 0 as ν gets smaller (larger threshold), because in such cases p_0 tends to 1.

We can compare the optimal randomised caching scheme with a scheme where only the K most popular files are always placed. In such case, denoted by Pop in the

figure, the hit probability is always

$$f_{pop} = (1 - p_0) \sum_{j=1}^{K} a_j = (1 - e^{-\nu}) \sum_{j=1}^{K} a_j. \tag{1.47}$$

Such policy which does not offer diversity among cache memories is strongly sub-optimal in case of multi-coverage, as the figure illustrates.

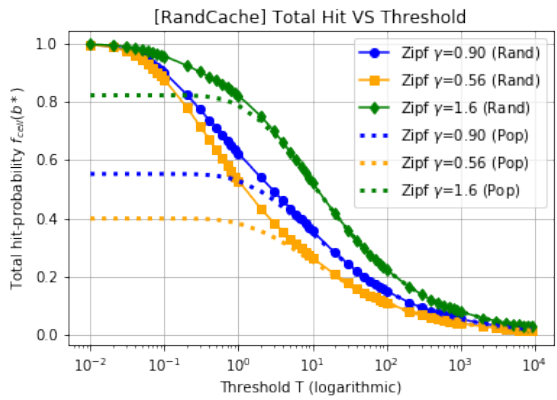

Figure 1.5 Evaluation of the optimal [`RandCache`] policy and comparison with the most popular policy [`Pop`] for the Boolean coverage model and various file popularity distributions.

1.5 Treating unequal file-sizes

The randomised content placement policy and the solution of `RandCache` have been based on the strong assumption that all files have equal size. Although one can argue that files can be split into chunks of equal size, this is not true in general. If we want to analyse more realistic cache systems, we will associate a tuple (a_j, z_j) per catalog item j, where z_j is the known file-size in *bytes*, e.g. a video can have size of 0.83 $[Gbyte]$. Consequently, the cache should have size K_b $[bytes]$, in contrast to the K $[files]$ used until this point. Then, we want to place content to caches, respecting the available memory size; the feasibility constraint (1.8) now takes the form

$$\sum_{j=1}^{F} X_{i,j} z_j \quad \leq \quad K_b. \tag{1.48}$$

This is a *binary Knapsack inequality*, and every randomised placement policy satisfies - similarly to (1.9) - the inequality

$$\mathbb{E}\left[\sum_{j=1}^{F} X_{i,j} z_j\right] = \sum_{j=1}^{F} b_j z_j \quad \leq \quad K_b, \tag{1.49}$$

where $b_j = \mathbb{P}(X_{i,j} = 1)$ is the placement probability (or frequency) of content j. We cannot use in a straightforward way the construction in Fig. 1.1 to sample batches, because these batches are not guaranteed to satisfy the Knapsack constraint!

The brute-force way would be to find the exhaustive set of feasible content batches

$$\mathscr{C} = \left\{ \mathbf{x} \mid \mathbf{x} \in \{0,1\}^F \,,\ \mathbf{x} \cdot \mathbf{b} \leq K \right\}. \tag{1.50}$$

From the above set we should further remove the feasible vectors $\tilde{\mathbf{x}} \in \mathscr{C}$, for whom there exists $\mathbf{x}' \in \mathscr{C}$, such that $\tilde{\mathbf{x}} \leq \mathbf{x}'$. The inequality is element-wise, meaning that $\tilde{x}_j \leq x'_j$, for all j. Then, if both $\mathscr{C}_1 = \{\text{“1”}\}$ and $\mathscr{C}_2 = \{\text{“1”}, \text{“4”}\}$ are feasible, we keep only the second batch, because otherwise memory space is left unused. We denote this more fine set by $\mathscr{C}^*$, and the feasible batches by $\mathscr{C}_\ell \in \mathscr{C}^*$, where the index runs over $\ell = 1, \ldots, L$. Here $L = card(\mathscr{C}^*)$.

We can now associate a frequency ρ_ℓ per batch, $\ell = 1, \ldots, L$, which satisfies

$$\sum_{\ell=1}^{L} \rho_\ell \quad = \quad 1. \tag{1.51}$$

The randomised policy would be the following: select a number within the interval $(0,1)$ uniformly at random, and pick the batch $k+1$, $k \in \{0, \ldots, L-1\}$, if the random number is within the interval $\left(\sum_{\ell=1}^{k} \rho_\ell, \sum_{\ell=1}^{k+1} \rho_\ell\right]$, with the convention $\sum_{\ell=1}^{0} \rho_\ell = 0$.

But one important detail remains open: how to determine the ρ_ℓ's from the given (or optimal) probabilities b_j's. We can use the fact that the frequency of object j is the sum of frequencies of batches where the b_j appears, i.e.,

$$b_j \quad = \quad \sum_{\ell=1}^{L} \rho_\ell \mathbf{1}_{\{j \in \mathscr{C}_\ell\}}, \quad j = 1, \ldots, F. \tag{1.52}$$

There are F such equalities and L unknowns. If we could solve the system $\mathbf{A}\rho = \mathbf{b}$, then we could determine the batch frequencies. Often, however $L > F$ or $L < F$ so the system is under- or over-determined. For such cases, we could use the Moore-Penrose pseudo-inverse, $\rho = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$, but this does not guarantee that the system actually has a solution.

In any case, an interesting observation from the brute-force method is that there is empty space in the cache with average size equal to

$$E \quad = \quad \sum_{\ell=1}^{L} \rho_\ell \left(K_b - \sum_{j=1}^{F} x_j^{(\ell)} z_j \right) \geq 0. \tag{1.53}$$

This observation actually motivates the following much simpler method.

1.5.1 Method: Overflow constraint

A more practical method is to replace the constraint (1.49) by the tighter constraint

$$\sum_{j=1}^{F} b_j z_j \quad \leq \quad K_b - \delta. \tag{1.54}$$

We saw previously that most feasible batches do not fill in the K_b memory completely. The above tightened inequality guarantees that the Knapsack constraint $\sum_{j=1}^{F} X_{i,j} z_j \leq K_b$ is violated only with some small probability ε. To see this,

$$\begin{aligned}\mathbb{P}\left(\sum_{j=1}^{F} X_{i,j} z_j > K_b\right) &\overset{Markov}{\leq} \frac{\mathbb{E}\left[\sum_{j=1}^{F} X_{i,j} z_j\right]}{K_b} = \frac{\sum_{j=1}^{F} b_j z_j}{K_b} \\ &\overset{(1.54)}{\leq} 1 - \frac{\delta}{K_b} = \varepsilon(\delta). \end{aligned} \tag{1.55}$$

We can also choose a desired violation bound ε, to determine the appropriate δ

$$\delta(\varepsilon) = (1-\varepsilon)K_b. \tag{1.56}$$

E.g., suppose we require a violation probability of 10%. Then $\varepsilon = 0.1$ and $\delta(0.1) = 0.9K_b$, so that the right-hand side of (1.54) is $0.1K_b$.

Now, we can use a similar construction as in Fig. 1.1, but without predefining the number of unit memories (it was K before). We place the b_j's one next to the other, say starting from $j = 1$ and on. If their sum at some point is larger than 1 then we split and move to the next line, until all F b_j's are placed. Then, similarly as in the random placement policy, we choose a number between 0 and 1 uniformly at random, draw a vertical line intersecting the above construction and we pick this batch of distinct contents. Note here, that now the number of contents can vary per batch. This method, guarantees that the batches are drawn so that $\mathbb{P}(X_{i,j} = 1) = b_j$. Remember that the b_j's where chosen such that the batches violate the memory constraint with probability ε.

1.5.2 [`RandCache`] with cache overflow constraint

We can now formulate and solve the `RandCache` problem, including the unequal file-sizes. We use the overflow constraint in (1.54), where $\delta(\varepsilon)$ is chosen based on the desired level of Knapsack violation, as in (1.56). The new feasible set of the random placement policy is

$$\mathscr{F}_{random}(\mathbf{z}, \delta) = \left\{ \mathbf{b} \mid \mathbf{b} \in [0,1]^F, \sum_{j=1}^{F} b_j z_j \leq K_b - \delta \right\}. \tag{1.57}$$

The problem takes the form

$$\begin{array}{lll} \max_{\mathbf{b}} & f_{cell}(\mathbf{b}) := 1 - \sum_{j=1}^{F} a_j \sum_{m=0}^{\infty} p_m (1-b_j)^m & \texttt{[Rand-z]} \\ \text{s.t} & \mathbf{b} \in \mathscr{F}_{random}(\mathbf{z}, \delta) & \end{array}. \tag{1.58}$$

In the above, the objective is the same as in the `RandCache` case, i.e. it is the average cache hit-probability. The file-sizes affect only the constraints.

This variation can be solved following the standard procedure with the tailored method we have presented for the `RandCache` problem. For completion, we show here the solution for `Rand-z`. Following the Lagrange relaxation, the primal vari-

ables are solved, similarly to Lemma 3. For fixed dual $\mu \geq 0$

$$b_j(\mu;z_j) \quad = \quad \begin{cases} 1, & \text{if } \frac{a_j}{z_j} p_1 > \mu \\ \omega(\mu;z_j), & \text{if } \frac{a_j}{z_j} p_1 \leq \mu \leq \frac{a_j}{z_j}\mathbb{E}[\mathscr{M}] \\ 0, & \text{if } \frac{a_j}{z_j}\mathbb{E}[\mathscr{M}] < \mu \end{cases} \tag{1.59}$$

where, now $\omega(\mu;z_j)$ is the solution over b_j of the equation

$$\frac{a_j}{z_j}\sum_{m=1}^{\infty} p_m m(1-b_j)^{m-1} \quad = \quad \mu. \tag{1.60}$$

Similarly, μ^* is the unique μ such that

$$b_1(\mu^*)z_1 + \ldots + b_F(\mu^*)z_F \quad = \quad K_b - \delta. \tag{1.61}$$

Structure of the solution: The difference compared to the equal (unit) size solution `RandCache` is subtle, and reveals interesting design insights for the optimal placement policy of `Rand-z`. When the file sizes **z** are taken into account, the primal solution takes the form (1.59). Compared to (1.35), it has the same structure, but one detail: the popularities a_j are replaced everywhere by the ratio a_j/z_j. Previously, we had ordered the files in decreasing order of popularity and optimally cached them with decreasing placement probability $b_j(\mu)$ following that content order. Now we should order the files by the ratio a_j/z_j and place them with decreasing probability $b_j(\mu;z_j)$ following the new order. So, this means that the size plays a major role in the placement decision. Even if a file is moderately popular, when its size is very large, it will be ordered low, and hence will have low placement probability. This is because, we want to avoid overflowing the cache memory. Instead, a very small-sized file with moderate popularity will be ordered high, and thus will be placed very frequently, because it is not so dangerous to overflow the cache.

1.6 Node mobility

A very common characteristic in cellular networks is node mobility. This is manifested as users moving from one cell to the other. In device-to-device (D2D) or vehicle-to-vehicle (V2V) communications, mobility plays a major role. User mobility results in loss of user association with a specific station, and a shift of service towards a different station nearby. Mobility requires an appropriate design of content placement policies. We follow the analysis in [5].

Suppose that a user requests content j at some time $t = 0$. This user is covered by a number of stations - say m. Till now, we considered the hit-probability, i.e. the probability that the content is cached in at least one covering station. But, the objective function in `RandCache` assumes silently that the typical user will not change position (and stick to its station association), until the file is completely downloaded, something not necessarily true. Suppose, instead, that the user does find the content cached in one of the covering stations, say station i, but at time $t = \tau$ she/he moves and the link is lost.

If during the interval $(0,\tau]$ the signal-to-noise ratio (SNR) with the associated station i is strong enough, then the whole file can been downloaded, i.e.,

$$\tau W \log_2 (1 + SNR(R_i)) \quad \geq \quad z_j. \tag{1.62}$$

The above uses the Shannon formula with bandwidth W. The $SNR(R_i) = SR_i^{-\beta}/W\sigma_N^2$, see also (1.14). If the inequality (1.62) is not fulfilled, then the file is not completely transmitted, so there is no hit. Hence,

$$\Psi_{i,j} \quad = \quad X_{i,j} \cdot \mathbf{1}_{\left\{\tau W \log_2(1+SNR(R_i)) \geq z_j\right\}} \in \{0,1\} \tag{1.63}$$

is the binary random variable for hit or miss, that takes into account not only content availability, but also file-size, user mobility and distance from the user. Let us consider station positions that follow a 2-dimensional Poisson Point Process (PPP) of density $\lambda \geq 0$, so there is an infinity of stations positioned on the plane, indexed by $i = 1,2,\ldots$. The hit probability that accounts for file download completion, is called here *service success probability*. It is the probability that at least one station i' satisfies $\Psi_{i',j} = 1$, so the objective now is

$$\begin{aligned} f_{d2d}(\mathbf{b}) &= \sum_{j=1}^{F} a_j \sum_{i=1}^{\infty} \mathbb{P}[\Psi_{i,j} \geq 1] \\ &= \sum_{j=1}^{F} a_j \left(1 - e^{-b_j \lambda h(\tau, z_j)}\right), \end{aligned} \tag{1.64}$$

where $h(\tau, z_j) = \pi\Gamma\left(\frac{3}{2}\right)\sqrt{S/W}\sigma_N^{-1}/\sqrt{2^{z_j/(W\tau)} - 1}$. The expression is taken from [5, Cor. 1] and refers to the case of Rayleigh fading and error exponent $\beta = 4$. The proof can also be found in the same reference.

The expression (1.64) satisfies the two properties from Lemma 1. It is separable in b_j. From the first and second partial derivatives, it is monotone increasing and concave in b_j. So the problem

$$\begin{array}{ll} \max_{\mathbf{b}} & f_{d2d}(\mathbf{b}) := \sum_{j=1}^{F} a_j \left(1 - e^{-b_j \lambda h(\tau, z_j)}\right) \quad \texttt{[Rand-D2D]} \\ \text{s.t} & \mathbf{b} \in \mathscr{F}_{random}(\mathbf{z}, \delta) \end{array} . \tag{1.65}$$

can be solved again using the method we have described for `RandCache`. We just mention that the derivative of the sub-problem `subD2D-j` is equal to

$$-\mu z_j + \frac{dg}{db_j} \quad = \quad -\mu z_j + a_j \lambda h(\tau, z_j) e^{-b_j \lambda h(\tau, z_j)}. \tag{1.66}$$

1.7 Competition among content providers

An interesting extension of the work, presented in [44, chapter 6] is when $S > 1$ content providers compete to share the same cache space and place their catalog. The providers are indexed by $s = 1,\ldots,S$ and each catalog has no overlap with any other, it is thus a disjoint set of contents. We index the contents of catalog s by $j_s = 1,\ldots,F_s$. The total memory space K_b per cache, will be split among the S

providers, and each will get a share k_s. The following inequality holds

$$\sum_{s=1}^{S} k_s \quad \leq \quad K_b. \tag{1.67}$$

Suppose that once k_s space is given to a provider, the catalog is placed following the random placement policy with placement variables $\mathbf{b}_s = (b_{s,1}, \ldots, b_{s,F_s})$. We get the following feasible set

$$\mathscr{F}_{rand,CP}(\mathbf{z}_1, \ldots, \mathbf{z}_S, \delta) = \left\{ \begin{array}{c} (\mathbf{b}_1, \ldots, \mathbf{b}_S) \mid \mathbf{b}_s \in [0,1]^{F_s}, \mathbf{k} \in [0, K_b]^S, \\ \sum_{s=1}^{S} k_s \leq K_b, \\ \sum_{j=1}^{F_s} b_{s,j} z_{s,j} \leq k_s - \delta, \ s = 1, \ldots, S \end{array} \right\}. \tag{1.68}$$

This set is characterised by $S+1$ inequalities, that is 1 inequality for the cache-space allocation and S inequalities for the content placement per allocated space k_s. This type of problem keeps the structure of the original `RandCache` problem, and its solution can be easily derived, using similar methods.

1.8 Beyond Independent Caches

The random independent content placement and the related `RandCache` formulation, although it has many advantages, it can still result in hit-probabilities much lower than those derived by detailed binary placement, both for a given realisation of user-station positions, e.g. from `BinCache`, as well as on average. The reason is that the caching decisions are done independently for different stations, so that two or more stations jointly covering a location may propose a complementary content only *by chance*. A possible way of improving upon this situation would consist in some content placement with cooperative decisions made between caches to guarantee content diversity among neighbouring nodes (see related works [45] and [46]).

The randomised policies can also be tuned in this direction by relaxing the assumption of independent placement. This was done in [47] where the dependence between caching decisions of different stations is described by a Gibbs distribution. Coming back to the setting of Section 1.1.1, with a given, finite configuration of stations, let us define the joint probability of placing the content at different stations according to the variables $\mathbf{x} \in \{0,1\}^{N \times F}$, (with $\sum_{j=1}^{F} x_{i,j} \leq K$ $(i = 1, \ldots, N)$) by

$$\pi_\beta(\mathbf{x}) := \frac{e^{\beta h(\mathbf{x})}}{Z_\beta}, \tag{1.69}$$

where

$$h(\mathbf{x}) = \int_{\mathscr{A}} \left[1 - \sum_{j=1}^{F} a_j \prod_{i \in \mathscr{H}(u)} (1 - x_{i,j}) \right] \mathrm{d}u$$

is the total hit probability in the deployment region $\mathscr{A}$ replacing the discrete sum over discrete user locations considered in (1.1), β is called the "inverse temperature" (motivated by literature from statistical Physics), and $Z_\beta = \sum_{\mathbf{x} \in \{0,1\}^{N \times F}} e^{\beta h(\mathbf{x})}$ is the normalizing constant called the "partition function".

Note that, $\lim_{\beta\to\infty}\sum_{\mathbf{x}\in\arg\max_{\mathbf{x}'\in\{0,1\}^{N\times F}} h(\mathbf{x}')} \frac{e^{\beta h(\mathbf{x})}}{Z_\beta} = 1$. Hence, if we choose a configuration $\mathbf{x}$ with probability $\pi_\beta(\mathbf{x})$, then, for sufficiently large β, the chosen configuration will be optimal (for the given configuration of stations, on average for different homogeneous user location in $\mathscr{A}$) with probability close to 1.

In most cases, there is no closed form expression for $\pi_\beta(\mathbf{x})$. Note however that we do not need to know probabilities $\pi_\beta(\mathbf{x})$. The only thing we need is to be able to sample from this distribution (so as to place the content according to the sampled values of $\mathbf{x}$ at different stations). And there are good ways of doing this. Indeed, there are several Gibbs sampling algorithms that simulate a discrete-time Markov chain on the space $\{0,1\}^{N\times F}$ having $\pi_\beta(\cdot)$ as the stationary probability distribution. In the particular case of our hit function $h(\mathbf{x})$ (which is called the Hamiltonian of the Gibbs distribution) these algorithms can be even made decentralized, in the sens that different stations can run them by updating their content knowing only the current choice of the content of the neighboring stations — i.e. those with which overlapping coverage regions. Moreover, the popularity can be learned during this process, and the parameter β can be appropriately increased to approach the optimal solution, a technique known as Simulated Annealing. We refer the reader to [47] for more details.

Observe that the above solution is proposed for a finite collection of stations. Gibbs models usually do not scale very well with the number of nodes (here stations) going to infinity. Mathematically challenging problems arise in this context, related to the long range propagation of the dependence between caching decisions of different stations. This problem was recently studied in [48] in a general mathematical context going beyond Gibbs distributions.

References

[1] Bastug E, Bennis M, Debbah M. Living on the edge: The role of proactive caching in 5G wireless networks. IEEE Communications Magazine. 2014;52(8):82–89.

[2] Shanmugam K, Golrezaei N, Dimakis AG, et al. FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers. IEEE Transactions on Information Theory. 2013;59(12):8402–8413.

[3] Wen J, Huang K, Yang S, et al. Cache-Enabled Heterogeneous Cellular Networks: Optimal Tier-Level Content Placement. IEEE Transactions on Wireless Communications. 2017;16(9):5939–5952.

[4] Jeon S, Hong S, Ji M, et al. Wireless Multihop Device-to-Device Caching Networks. IEEE Transactions on Information Theory. 2017;63(3):1662–1676.

[5] Jarray C, Giovanidis A. Successful file transmission in mobile D2D networks with caches. Computer Networks. 2018;147:162 – 179.

[6] Kurose J. Information-centric networking: The evolution from circuits to packets to content. Computer Networks. 2014;66:112 – 120. Leonard Kleinrock Tribute Issue: A Collection of Papers by his Students.

[7] Garetto M, Leonardi E, Martina V. A Unified Approach to the Performance Analysis of Caching Systems. ACM Trans Model Perform Eval Comput Syst. 2016 May;1(3).

[8] Jiang B, Nain P, Towsley D. On the Convergence of the TTL Approximation for an LRU Cache under Independent Stationary Request Processes. ACM Trans Model Perform Eval Comput Syst. 2018 Sep;3(4).

[9] Giovanidis A, Avranas A. Spatial Multi-LRU Caching for Wireless Networks with Coverage Overlaps. SIGMETRICS Perform Eval Rev. 2016 Jun;44(1):403?405.

[10] Poularakis K, Iosifidis G, Tassiulas L. Approximation Algorithms for Mobile Data Caching in Small Cell Networks. IEEE Transactions on Communications. 2014;62(10):3665–3677.

[11] Krolikowski J, Giovanidis A, Di Renzo M. A Decomposition Framework for Optimal Edge-Cache Leasing. IEEE Journal on Selected Areas in Communications. 2018;36(6):1345–1359.

[12] Blaszczyszyn B, Giovanidis A. Optimal geographic caching in cellular networks. In: 2015 IEEE International Conference on Communications (ICC); 2015. p. 3358–3363.

[13] Chae SH, Choi W. Caching Placement in Stochastic Wireless Caching Helper Networks: Channel Selection Diversity via Caching. IEEE Transactions on Wireless Communications. 2016;15(10):6626–6637.

[14] Cui Y, Jiang D. Analysis and Optimization of Caching and Multicasting in Large-Scale Cache-Enabled Heterogeneous Wireless Networks. IEEE Transactions on Wireless Communications. 2017;16(1):250–264.

[15] Li K, Yang C, Chen Z, et al. Optimization and Analysis of Probabilistic Caching in N -Tier Heterogeneous Networks. IEEE Transactions on Wireless Communications. 2018;17(2):1283–1297.

[16] Serbetci B, Goseling J. On Optimal Geographical Caching in Heterogeneous Cellular Networks. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC); 2017. p. 1–6.

[17] Liu D, Yang C. Caching Policy Toward Maximal Success Probability and Area Spectral Efficiency of Cache-Enabled HetNets. IEEE Transactions on Communications. 2017;65(6):2699–2714.

[18] Wen W, Cui Y, Zheng F, et al. Random Caching Based Cooperative Transmission in Heterogeneous Wireless Networks. IEEE Transactions on Communications. 2018;66(7):2809–2825.

[19] Zhao J, Zhao S, Qu H, et al. Analysis and Optimization of Probabilistic Caching in Micro/Millimeter Wave Hybrid Networks With Dual Connectivity. IEEE Access. 2018;6:72372–72380.

[20] Zhu Y, Zheng G, Wang L, et al. Content Placement in Cache-Enabled Sub-6 GHz and Millimeter-Wave Multi-Antenna Dense Small Cell Networks. IEEE Transactions on Wireless Communications. 2018;17(5):2843–2856.

[21] Chen Z, Pappas N, Kountouris M. Probabilistic Caching in Wireless D2D Networks: Cache Hit Optimal Versus Throughput Optimal. IEEE Communications Letters. 2017;21(3):584–587.

[22] Malak D, Al-Shalash M, Andrews JG. Spatially Correlated Content Caching for Device-to-Device Communications. IEEE Transactions on Wireless Communications. 2018;17(1):56–70.

[23] Lin X, Xia J, Wang Z. Probabilistic caching placement in UAV-assisted heterogeneous wireless networks. Physical Communication. 2019;33:54 – 61.

[24] Song F, Li J, Ding M, et al. Probabilistic Caching for Small-Cell Networks With Terrestrial and Aerial Users. IEEE Transactions on Vehicular Technology. 2019;68(9):9162–9177.

[25] Emara M, Elsawy H, Sorour S, et al. Optimal Caching in 5G Networks With Opportunistic Spectrum Access. IEEE Transactions on Wireless Communications. 2018;17(7):4447–4461.

[26] Gao J, Zhang S, Zhao L, et al. The Design of Dynamic Probabilistic Caching with Time-Varying Content Popularity. IEEE Transactions on Mobile Computing. 2020;p. 1–1.

[27] Krishnan S, Afshang M, Dhillon HS. Effect of Retransmissions on Optimal Caching in Cache-Enabled Small Cell Networks. IEEE Transactions on Vehicular Technology. 2017;66(12):11383–11387.

[28] Yang Q, Wang H, Zheng T. Delivery-Secrecy Tradeoff for Cache-Enabled Stochastic Networks: Content Placement Optimization. IEEE Transactions on Vehicular Technology. 2018;67(11):11309–11313.

[29] Liu D, Yang C. A Learning-Based Approach to Joint Content Caching and Recommendation at Base Stations. In: 2018 IEEE Global Communications Conference (GLOBECOM); 2018. p. 1–7.

[30] Choi M, Kim J, Moon J. Wireless Video Caching and Dynamic Streaming Under Differentiated Quality Requirements. IEEE Journal on Selected Areas in Communications. 2018;36(6):1245–1257.

[31] Avrachenkov K, Goseling J, Serbetci B. A Low-Complexity Approach to Distributed Cooperative Caching with Geographic Constraints. Proc ACM Meas Anal Comput Syst. 2017 jun;1(1).

[32] Baccelli F, Błaszczyszyn B. Stochastic Geometry and Wireless Networks: Volume I Theory. Foundations and Trends in Networking. 2010;3(3-4):249–449.

[33] Andrews JG, Baccelli F, Ganti RK. A Tractable Approach to Coverage and Rate in Cellular Networks. IEEE Transactions on Communications. 2011;59(11):3122–3134.

[34] Keeler HP, Błaszczyszyn B, Karray MK. SINR-based k-coverage probability in cellular networks with arbitrary shadowing. In: 2013 IEEE International Symposium on Information Theory; 2013. p. 1167–1171.

[35] Błaszczyszyn B, Haenggi M, Keeler P, et al. Stochastic geometry analysis of cellular networks. Cambridge University Press; 2018.

[36] Dhillon HS, Ganti RK, Baccelli F, et al. Modeling and Analysis of K-Tier Downlink Heterogeneous Cellular Networks. IEEE Journal on Selected Areas in Communications. 2012;30(3):550–560.

[37] Keeler HP. SINR-based k-coverage probability in cellular networks (https://www.mathworks.com/matlabcentral/fileexchange/40087-sinr-

based-k-coverage-probability-in-cellular-networks). MATLAB Central File Exchange. Retrieved September 7, 2020;.

[38] Dan A, Towsley D. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. In: Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems. SIGMETRICS '90. New York, NY, USA: Association for Computing Machinery; 1990. p. 143?152.

[39] Newman MEJ. Power laws, Pareto distributions and Zipf's law. Contemporary Physics 46, pp 323351. 2005;.

[40] Breslau L, Pei Cao, Li Fan, et al. Web caching and Zipf-like distributions: evidence and implications. In: IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320). vol. 1; 1999. p. 126–134 vol.1.

[41] Boyd S, Vandenberghe L. Convex Optimization. Cambridge University Press; 2004.

[42] Bertsekas DP. Network Optimization: continuous and discrete models. Athena Scientific; 1998.

[43] Boyd S. Subgradient Methods. Nortes for EE364b, Stanford University. 2014;.

[44] Krolikowski J. Optimal Content Management and Dimensioning in Wireless Networks. PhD Thesis Networking and Internet Architecture [csNI], NNT : 2018SACLS452, tel-02124294. 2018;.

[45] Malak D, Al-Shalash M, Andrews JG. Spatially Correlated Content Caching for Device-to-Device Communications. IEEE Transactions on Wireless Communications. 2018;17(1):56–70.

[46] Malak D, Mdard M, Yeh EM. Spatial Soft-Core Caching. In: 2019 IEEE International Symposium on Information Theory (ISIT); 2019. p. 2009–2013.

[47] Chattopadhyay A, Błaszczyszyn B, Keeler HP. Gibbsian on-line distributed content caching strategy for cellular networks. IEEE Transactions on Wireless Communications. 2017;17(2):969–981.

[48] Błaszczyszyn B, Hirsch C. Optimal stationary markings. arXiv preprint arXiv:200108074. 2020;.