Queueing Systems manuscript No.
(will be inserted by the editor)

Transient and Steady-state Regime of a Family of List-based Cache Replacement Algorithms

Nicolas Gast · Benny Van Houdt

the date of receipt and acceptance should be inserted later

Abstract We study the performance of a family of cache replacement algorithms. The cache is decomposed into lists. Some of these lists can be virtual in the sense that only meta-data is stored in those lists. An item enters the cache via the first list and jumps to the next list whenever a hit on it occurs. The classical policies FIFO, RANDOM, CLIMB and its hybrids are obtained as special cases. We present explicit expressions for the cache content distribution and miss probability under the IRM model. We develop an algorithm with a time complexity that is polynomial in the cache size and linear in the number of items to compute the exact miss probability. We introduce lower and upper bounds on the latter that can be computed in a time that is linear in the cache size times the number of items.

We introduce a mean field model to approximate the transient behavior of the miss probability and prove that this model becomes exact as the cache size and number of items go to infinity. We show that the set of ODEs associated to the mean field model has a unique fixed point that can be used to approximate the miss probability in case the exact computation is too time consuming.

Using this approximation, we provide guidelines on how to select a replacement algorithm within the family considered such that a good trade-off is achieved between the cache reactivity and its steady-state hit probability. We simulate these cache replacement algorithms on traces of real data and

This paper is an extended version of a conference paper [29]. The key addition of this journal version is the introduction of the notion of virtual lists in which only meta-data is stored.

Nicolas Gast
Inria
Univ. Grenoble Alpes
CNRS, LIG, F-38000 Grenoble, France
E-mail: nicolas.gast@inria.fr

B. Van Houdt
Department of Mathematics and Computer Science
University of Antwerp - iMinds, Belgium
E-mail: benny.vanhoudt@uantwerpen.be

show that they can outperform LRU. Finally, we also disprove the well-known conjecture that the CLIMB algorithm is the optimal finite-memory replacement algorithm under the IRM model.

Keywords Cache Replacement Policies; Independent Reference Model; Storage Management; Self-organizing lists; Miss-ratio

1 Introduction

Caches are omnipresent in modern day computer systems to improve system performance. A plethora of cache replacement algorithms has been studied by various authors over the past few decades. Nevertheless, explicit expressions for the cache content distribution and miss probability under the well-known Independent Reference Model (IRM) are only available for a limited number of replacement algorithms such as FIFO, RANDOM, LRU, CLIMB and some simple hybrids thereof. Further, these results are of little practical use due to the curse of dimensionality (except for FIFO and RANDOM). As such many approximation methods have been proposed for single caches [9,8,13,28,5] as well as for networks of caches [25,11,30,14,28,5]. While these approximations have often been shown to result in highly accurate predictions for the miss probability (under the IRM model), their theoretical support is most often rather limited. Further, these prior studies typically focus on the long term (steady state) behavior only, while the transient behavior of the miss probability is also of interest under more dynamic request patterns.

In this paper we study two broad classes of cache replacement algorithms that both organize the cache content in a number of lists (with a fixed size). Items enter the cache via the first list and are promoted to the next list whenever a hit occurs at the expense of demoting another item. Some of these lists can be *virtual*, meaning that only meta-data is stored for the items in those lists. The difference between the two classes of replacement algorithms exists in the manner in which the demoted item is selected. These two classes of cache replacement algorithms contain the well-known RANDOM, FIFO and CLIMB algorithm as well as some more advanced algorithms introduced in [2] as special cases.

The main contributions of the paper are as follows:

1. We show that both classes of algorithms perform alike under the IRM model by deriving an explicit expression for the steady-state cache content distribution and miss probability that is valid in both classes. We subsequently exploit this expression to devise fast algorithms to compute the exact overall and per-item miss probability, thereby avoiding the traditional curse of dimensionality. The time complexity to compute the overall miss probability is linear in the number of items and polynomial in the cache size (where the degree is equal to the number of lists used). We also derive upper and lower bounds on the miss probability that can be computed in a time that is linear in the cache size.

2. We introduce a mean field model (for one of the two classes of replacement algorithms) that can be used to approximate the transient behavior of the miss probability under the IRM model and show that this model becomes exact as the number of items and cache size becomes infinitely large (under some mild conditions). We prove that the set of ODEs that characterizes the mean field model has a unique fixed point and demonstrate that this point can be used to get a highly accurate approximation of the miss probability whenever the exact computation becomes too time consuming.
3. We provide a number of guidelines on how to select the number of lists and list sizes by relying on various experiments using both the IRM model and trace-based simulations. These experiments demonstrate that the algorithms considered in this paper can outperform LRU. We also show that the well-known conjecture [2, page 135] that the CLIMB algorithm is the optimal finite-memory demand replacement algorithm under the IRM model is false.

The paper is structured as follows. We describe related work in Section 2. In Section 3, we describe the two classes of replacement algorithms. We derive the exact steady-state probabilities and miss probabilities as well as upper and lower bounds in Section 4. We develop our mean-field approximation and show its validity in Section 5. Finally, we provide empirical evidence on how the list sizes should be chosen in Section 6. We conclude in Section 7.

2 Related Work

Cache replacement algorithms have been analyzed by various authors mostly under the well-known IRM model. While this model is not very suitable in the context of secondary memory management, it is argued by many authors to be a reasonable model in a web caching context. For instance, as stated in [7], for some purposes, one might model web accesses by a simple model that assumes independent references following a Zipf-like distribution and no correlation between request frequency and item size.

Explicit results under the IRM model for the cache content distribution and miss probability have been derived for FIFO and LRU [23, 2], RANDOM [16] and some simple hybrids thereof [1, 3]. LRU is shown to outperform FIFO in [34], while FIFO and RANDOM perform identical [16]. These expressions however can only be used directly to compute the miss probability of very small caches (e.g., 20 items). For FIFO, an algorithm to compute the exact miss probability in $O(mn)$ time is given in [10], where n is the number of items and m the cache size.

A number of approximations have been proposed to analyze caches of reasonable size (they are fast to compute and can be used for cache sizes of more than millions of items). In [9] a fast iterative scheme is proposed for FIFO that coincides with the iterative algorithm proposed in [28] for RANDOM. For LRU the author of [9] introduce an $O(mn)$ time approximation, which is of limited

use since the introduction of what is now called the Che-approximation [8], for which theoretical support is provided in [13].

Recently, the idea of the Che-approximation was generalized to capture a broader class of replacement algorithms, under the IRM model, in case of a renewal model [28] and for general MAP arrivals [15]. While [28] also considers replacement policies that make use of multiple lists, all but one of these lists are assumed to be virtual lists that only store the meta-data and not the actual items. Further, the algorithms are only approximation, but no proof of accuracy is provided. The analysis of [15] complements our paper and provides a provable approximation that is asymptotically exact (as opposed to the one of [28]). Renewal models were also considered in [22,11,5] for time-to-live caches which, in light of the Che-approximation, can be used to approximate capacity-driven replacement polices. Another line of work is to consider that the popularities of item vary with time. As shown in [32], this can be modeled by a shot-noise model that describes the temporal locality between consecutive requests. Che-like approximation for such models are developed in [26].

More practically oriented simulation based studies have also been performed and we do not attempt to provide an overview here. We would like to indicate that the idea of using multiple lists has been explored before, e.g., the 2Q and LIRS policies both combat the poor performance of LRU on *scan sequences* and typical database access patterns [21,20].

Considerable efforts have also been made to develop approximation methods to analyze networks of caches [25,11,30,14,28,5]. Some of these methods are iterative in nature and require the repeated solution of isolated caches, [30]. Hence, studying isolated caches is useful in the study of networks of caches.

Fluid limits for the miss probability have been derived for LRU [19], 2Q [18] and RANDOM [33]. Our mean field result is more general compared to [18,33]. These papers consider a system with n classes of items and N items per class. The items within a class are indistinguishable (they all have the same popularity). They show that their approximation is correct as N goes to infinity. Our approach is more general since we do not assume that items can be clustered into a finite number of classes. Rather, our bounds in 6 depends directly on the popularity of the most popular objects p_i but not on the number of objects.

3 Replacements algorithms

In this section we introduce two families of replacement algorithms that are obtained by slightly modifying a family of replacement algorithms introduced in [2]. We denote the members of these families as FIFO($\mathbf{m}, v$) and RAND($\mathbf{m}, v$), respectively, where $\mathbf{m} = (m_1, m_2, \ldots, m_h)$ is a tuple of integers with $m_i \geq 1$ (for $i = 1, \ldots, h$) and $v < h$.

The FIFO($\mathbf{m}, v$) replacement algorithm: The FIFO($\mathbf{m}, v$) algorithm makes use of $h > v$ lists, labeled 1 to h. Only the items present in list $v+1$ to h are stored

in the cache, while for the items in lists 1 to v only meta-data is stored. As in [28], we say that lists 1 to v are *virtual* lists. The length of list i is equal to m_i, for $i = 1, \ldots, h$, and the total cache size is equal to $m = \sum_{i=v+1}^{h} m_i$. Items enter the set of lists via list 1 and whenever requested while being part of list i they move up one list. More specifically, one of the following four events occurs when an item, say item k, is requested at some point in time:

1. Item k was not in any of the lists (miss) – in this case item k is inserted in position 1 of list 1. The remaining items in list 1 move back one position, while the item that was in the last position of list 1, that is, in position m_1, is removed from the cache.
2. Item k was in position j of list $i \leq v$ (miss) – in this case item k moves to position 1 of list $i+1$, all the items in list $i+1$ move back one position, the item that was in position m_{i+1} of list $i+1$ takes the former position of item k, that is, position j in list i.
3. Item k was in position j of list $v < i < h$ (hit) – this case is identical to the previous except that there is a hit.
4. Item k was part of list h (hit) – in this case no changes are made.

The RAND($\mathbf{m}, v$) replacement algorithm: The RAND($\mathbf{m}, v$) algorithm operates in a manner similar to the FIFO($\mathbf{m}, v$) algorithm, except for two changes. First, when the requested item is not part of any list, it is inserted in a *random* position in list 1 (as opposed to position 1) and the item that was in this randomly selected position is removed from the cache. Second, when a request occurs for an item part of list $i < h$, say in position j, the item is inserted in a *random* position in list $i+1$, while the item that was in this randomly selected position moves to position j of list i. In other words, only two items change position in case a hit in list $i < h$ occurs.

We will show that the FIFO($\mathbf{m}, v$) and RAND($\mathbf{m}, v$) algorithm have the same steady state distribution for the cache content under the IRM model, which is a generalization of the well-known fact that FIFO and RANDOM perform alike under the IRM model [16]. In fact, one can even show that the steady state probabilities of FIFO($\mathbf{m}, v$) remain the same if we do not update the lists when an item part of some lists is requested unless the item is in the first position of some list i, for $i = 1, \ldots, h-1$.

A natural extension of these policies is to consider the two following variants: strict FIFO($\mathbf{m}, v$) and LRU($\mathbf{m}, v$):

- *strict FIFO($\mathbf{m}, v$)* – The only difference between FIFO($\mathbf{m}$,v) and strict FIFO($\mathbf{m}, v$) is when a request occurs for an item that is in position j of a list $i < h$. As before, item k moves to position 1 of list $i+1$ and all the items in list $i+1$ move back one position. The difference is that the item that was in position m_{i+1} of list $i+1$ moves to position 1 of list i and the items that were in position 1 to $j-1$ of list i move back one position.
- *LRU($\mathbf{m}, v$)* – The difference between LRU($\mathbf{m}, v$) and strict FIFO($\mathbf{m}, v$) is when a hit occurs in position j of list h. In this case, this item is moved to

the first position of list h and all the items in the positions 1 to $j-1$ of list h move back one position.

The strict FIFO$(\mathbf{m}, 0)$ algorithm corresponds to the policy $\mathcal{A}^h_{m_h}(m_h, \ldots, m_1)$ of [2], while the LRU$(\mathbf{m}, 0)$ algorithm is denoted as $\mathcal{A}^h_1(m_h, \ldots, m_1)$ in [2]. Note that the lists in [2] are labeled in the opposite order. The FIFO$(\mathbf{m}, v)$ and the strict FIFO$(\mathbf{m}, v)$ algorithms operate identical when $m_1 = \ldots = m_{h-1} = 1$. For this special case, closed-form results for the steady-state probabilities for FIFO$(\mathbf{m}, 0)$ algorithm under the IRM model were also derived in [2].

Although these algorithms are close to our definition of FIFO$(\mathbf{m}, v)$, simple closed-form expressions for the cache content distribution and miss probabilities do not appear to exist. In Section 6, we compare all these policies using trace-based simulation. We demonstrate that the hit probabilities of strict FIFO$(\mathbf{m}, 0)$ and FIFO$(\mathbf{m}, 0)$ are close to each other (unless m_1 is close to m), while LRU$(\mathbf{m}, 0)$ outperforms these policies only marginally (at the expense of slightly more work).

4 Steady state probabilities

In this section, we compute the stationary measure of the policies FIFO$(\mathbf{m}, v)$ and RAND$(\mathbf{m}, v)$ under the independent reference model (IRM). This models means that the sequence of requests is a sequence of *i.i.d.* random variables. We denote by p_i the probability the id of a requested item it i.

4.1 The product-form stationary measure

Let $Y_{i,j}(t)$, with $(i,j) \in \mathcal{I}(\mathbf{m}) \triangleq \{(i,j) | i = 1, \ldots, h, j = 1, \ldots, m_i\}$, be the id of the item in position j of list i at time t, where we observe the set of lists whenever a request arrives. Let $\mathcal{C}_n(m)$ consists of all the sequences $(c_1, \ldots, c_m)$ of m distinct integers taken from the set $\{1, \ldots, n\}$. Under the IRM model the process $\mathbf{Y} = \{(Y_{i,j}(t), (i,j) \in \mathcal{I}(\mathbf{m})), t \geq 0\}$ is clearly a Markov chain on the state space $\mathcal{C}_n(m)$ for the FIFO$(\mathbf{m}, v)$ and RAND$(\mathbf{m}, v)$ algorithms. Denote $\pi_{\mathrm{A}}(\mathbf{c})$, with $\mathbf{c} = (c_1, \ldots, c_m)$, as the steady state probability of state $\mathbf{c}$, where A = FIFO$(\mathbf{m}, v)$ or RAND$(\mathbf{m}, v)$ denotes the replacement algorithm used. Note that the evolution of the Markov chain does not depend on the parameter v.

To ease the notation denote $c_{\sum_{s=1}^{i-1} m_s + j}$ as $c(i,j)$ for $(i,j) \in \mathcal{I}(\mathbf{m})$, where we can think of $c(i,j)$ as the id of the item in position j at list i. The next theorem shows that the steady state probabilities of this Markov chain have a simple closed form for the FIFO(**m**,v) or RAND(**m**,v) algorithm:

Theorem 1 *The steady state probabilities $\pi_{RAND(\mathbf{m},v)}(\mathbf{c})$ and $\pi_{FIFO(\mathbf{m},v)}(\mathbf{c})$, with $\mathbf{c} \in \mathcal{C}_n(m)$, can be written as*

$$\pi_{FIFO(\mathbf{m},v)}(\mathbf{c}) = \pi_{RAND(\mathbf{m},v)}(\mathbf{c}) = \pi(\mathbf{c}) \triangleq \frac{1}{Z(\mathbf{m})} \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{c(i,j)} \right)^i, \qquad (1)$$

where $Z(\mathbf{m}) = \sum_{\mathbf{c}\in\mathcal{C}_n(m)} \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{c(i,j)}\right)^i$.

Proof The proof consists in verifying the balance equations. We start with the RAND$(\mathbf{m}, v)$ algorithm. Denote $\mathbf{c}_{(i,j)\leftrightarrow(i',j')}$ as the vector $\mathbf{c}$ with $c(i,j)$ and $c(i',j')$ exchanged and $\mathbf{c}_{k\to(i,j)}$ as the vector $\mathbf{c}$ with entry $c(i,j)$ replaced by k. We can express the global balance equation of state $\mathbf{c}$ as

$$\pi_{\mathrm{RAND}(\mathbf{m},v)}(\mathbf{c})\left(1-\sum_{j=1}^{m_h} p_{c(h,j)}\right) = \sum_{k\neq c_1,\ldots,c_m}\sum_{u=1}^{m_1} \pi_{\mathrm{RAND}(\mathbf{m},v)}(\mathbf{c}_{k\to(1,u)})\frac{p_{c(1,u)}}{m_1} + \sum_{i=1}^{h-1}\sum_{u=1}^{m_i}\sum_{w=1}^{m_{i+1}} \pi_{\mathrm{RAND}(\mathbf{m},v)}(\mathbf{c}_{(i,u)\leftrightarrow(i+1,w)})\frac{p_{c(i+1,w)}}{m_{i+1}},$$

as we exit state $\mathbf{c}$ unless there is a hit on one of the entries in list h, while we can enter state $\mathbf{c}$ because there is a miss (and some item is inserted in a random position in list 1) or there is a request for an item in list i that moves up to a random position at list $i+1$. Implicitly plugging in (1) and noting that $\pi(\mathbf{c}_{(i,u)\leftrightarrow(i+1,v)})/\pi(\mathbf{c}) = p_{c(i,u)}/p_{c(i+1,v)}$ and $\pi(\mathbf{c}_{k\to(1,u)})/\pi(\mathbf{c}) = p_k/p_{c(1,u)}$ holds for (1), yields

$$1-\sum_{j=1}^{m_h} p_{c(h,j)} = \sum_{k\neq c_1,\ldots,c_m} p_k + \sum_{i=1}^{h-1}\sum_{u=1}^{m_i} p_{c(i,u)},$$

which clearly holds as $\sum_{i=1}^{n} p_i = 1$.

For FIFO$(\mathbf{m}, v)$, let the vector $\mathbf{c}_{\mathrm{miss}(k)}$ be identical to $\mathbf{c}$ except that the entries $(c_{\mathrm{miss}(k)}(1,1),\ldots,c_{\mathrm{miss}(k)}(1,m_1))$ are equal to $(c(1,2),\ldots,c(1,m_1),k)$ and denote $\mathbf{c}_{\mathrm{hit}(i,j)}$ as the vector $\mathbf{c}$ except that $(c_{\mathrm{hit}(i,j)}(i+1,1),\ldots,c_{\mathrm{hit}(i,j)}(i+1,m_{i+1})) = (c(i+1,2),\ldots,c(i+1,m_1),c(i,j))$ and $c_{\mathrm{hit}(i,j)}(i,j) = c(i+1,1)$.

$$\pi_{\mathrm{FIFO}(\mathbf{m},v)}(\mathbf{c})\left(1-\sum_{j=1}^{m_h} p_{c(h,j)}\right) = \sum_{k\neq c_1,\ldots,c_m} \pi_{\mathrm{FIFO}(\mathbf{m},v)}(\mathbf{c}_{\mathrm{miss}(k)})p_{c(1,1)} + \sum_{i=1}^{h-1}\sum_{j=1}^{m_i} \pi_{\mathrm{FIFO}(\mathbf{m},v)}(\mathbf{c}_{\mathrm{hit}(i,j)})p_{c(i+1,1)}.$$

The result follows by noting that $\pi(\mathbf{c}_{\mathrm{miss}(k)})/\pi(\mathbf{c}) = p_k/p_{c(1,1)}$ and $\pi(\mathbf{c}_{\mathrm{hit}(i,j)})/\pi(\mathbf{c}) = p_{c(i,j)}/p_{c(i+1,1)}$ holds for (1). □

When $h = 1, v = 0$ and $\mathbf{c} = (c_1,\ldots,c_m)$ we get

$$\pi(\mathbf{c}) = \frac{1}{Z(\mathbf{m})}\prod_{j=1}^{m} p_{c_j}.$$

These are the well-known steady state probabilities for the FIFO and RANDOM algorithm [23,16]. When $h = m$, meaning m_i=1 for all i, the RAND$(\mathbf{m}, 0)$

algorithm coincides with the so-called TRANSPOSITION rule or CLIMB algorithm [31,17]. In this case the above theorem reduces to

$$\pi(\mathbf{c}) = \frac{1}{Z(\mathbf{m})} \prod_{j=1}^{m} p_{c_j}^j,$$

which are the steady state probabilities of the CLIMB algorithm as reported in [2, Section 4.2]. Theorem 1 also generalizes the results for the second family of replacement algorithms studied in [1] which corresponds to $m_1{=}\ldots{=}m_{h-1}{=}1$ and $v = 0$

$$\pi(\mathbf{c}) = \frac{1}{Z(\mathbf{m})} \prod_{j=1}^{m} p_{c_j}^{\min(j,h)}.$$

Due to Theorem 1 it is easy to check that the following corollary holds:

Corollary 1 *For the RAND($\mathbf{m}, v$) replacement algorithm the Markov chain* $\mathbf{Y}$ *is reversible,* i.e.*,*

$$\pi(\mathbf{c})\mathbf{P}[Y(t{+}1) = \mathbf{c}'|Y(t){=}\mathbf{c}] = \pi(\mathbf{c}')\mathbf{P}[Y(t{+}1) = \mathbf{c}|Y(t){=}\mathbf{c}'],$$

for any $\mathbf{c}, \mathbf{c}' \in \mathcal{C}_n(m)$.

Denote the hit probability of the RAND$(\mathbf{m}, v)$ and FIFO$(\mathbf{m}, v)$ algorithms as $H(\mathbf{m}, v)$ and let $M(\mathbf{m}, v) = 1 - H(\mathbf{m}, v)$ be the miss probability under the IRM model. Clearly, one can express the miss probability via the steady state probabilities

$$M(\mathbf{m}, v) = \sum_{\mathbf{c}\in\mathcal{C}_n(m)} \left(1 - \sum_{i=v+1}^{h} \sum_{j=1}^{m_i} p_{c(i,j)}\right) \pi(\mathbf{c}). \tag{2}$$

This formula is however not very useful to compute $M(\mathbf{m}, v)$, unless m is very small, as the number of terms is exponential in m. In the next two sections we indicate how to compute the overall and per item miss probability in a more efficient manner.

4.2 Exact overall miss probability

In this section we introduce Algorithm 1 that computes the steady-state miss probability in a time that is polynomial in the cache size m, where the degree of the polynomial is equal to h. This algorithm is a generalization of the methodology introduced in [10] for the FIFO algorithm (which corresponds to the case with $h = 1$). It uses a dynamic programming approach: to compute $M(\mathbf{m}, v)$, for $v < h$, we compute $M(\mathbf{m}', v)$ for all $\mathbf{m}' \leq \mathbf{m}$ (component-wise). In other words, the quantities $M(\mathbf{m}', v)$ are obtained as by-products of the

computation of $M(\mathbf{m}, v)$. Obtaining these quantities is useful when studying optimal list sizes. More precisely, define a first set of list sizes

$$\mathcal{V}_h^{\leq m} = \left\{ (m_1 \dots m_k) \middle| k \leq h, \sum_{i=1}^{k} m_i \leq m \right\},$$

and a second set of list sizes

$$\mathcal{V}_h^{m_1 \dots m_h} = \{ (m'_1 \dots m'_k) \mid k \leq h, m'_i \leq m_i \}.$$

Our algorithm computes the miss probabilities $M(\mathbf{m}, v)$ for all $\mathbf{m} \in \mathcal{V}_h^{\leq m}$ in $O(nh^2 m^h / h!)$ time and for $\mathbf{m} \in \mathcal{V}_h^{m_1 \dots m_h}$ in $O(nh^2 \prod_{i=1}^{h}(m_i + 1))$ time. In the next section, we show how to compute the corresponding per item hit probabilities in $O(n^{3/2} h^2 m^h / h!)$ and $O(n^{3/2} h^2 \prod_{i=1}^{h}(m_i + 1))$ time, respectively.

Let $p_1, \dots, p_n$ be a *fixed* but arbitrary ordering of the request probabilities and $\mathbf{r} = (r_1, \dots, r_h)$ with r_i integer for all i and $r = \sum_i r_i$. Let $\mathbf{e}_j$ be the j-th row of the size h identity matrix. Define $E(\mathbf{0}, k) = 1$, $E(\mathbf{r}, k) = 0$ if $\sum_i r_i > k$ or $r_j < 0$ for some j and

$$E(\mathbf{r}, k) = \sum_{\mathbf{c} \in \mathcal{C}_k(r)} \prod_{i=1}^{h} \left(\prod_{j=1}^{r_i} p_{c(i,j)} \right)^i. \tag{3}$$

By noting that the probability that the requested item is in position j of list i is the same for any $j \leq m_i$ and using (2) we can express $M(\mathbf{m}, v)$ via $E(\mathbf{r}, k)$:

$$\begin{aligned} M(\mathbf{m}, v) &= \sum_{\mathbf{c} \in \mathcal{C}_n(m)} \left(\sum_{k \notin \mathbf{c}} p_k + \sum_{i=1}^{v} \sum_{j=1}^{m_i} p_{c(i,j)} \right) \pi(\mathbf{c}) \\ &= \frac{E(\mathbf{m} + \mathbf{e}_1, n) + \sum_{i=1}^{v} m_i E(\mathbf{m} + \mathbf{e}_{i+1} - \mathbf{e}_i, n)}{E(\mathbf{m}, n)}. \end{aligned}$$

Further, we have the following recursion for $E(\mathbf{r}, k)$ by noting that item k appears at most once in $\mathbf{c}$:

$$E(\mathbf{r}, k) = E(\mathbf{r}, k-1) + \sum_{j=1}^{h} r_j p_k^j E(\mathbf{r} - \mathbf{e}_j, k-1). \tag{4}$$

In principle we can use this recursion with $E(\mathbf{e}_j, 1) = p_1^j$ to compute the miss probabilities. However, $E(\mathbf{r}, k)$ decreases quickly in $\sum_i r_i$ and can easily cause underflows even for m values below 100 (as in [10] for the FIFO replacement policy). To avoid this problem, we define for $i = 1, \dots, h$:

$$F_i(\mathbf{r}, k) = \frac{E(\mathbf{r}, k)}{E(\mathbf{r} - \mathbf{e}_i, k)}, \tag{5}$$

for $\mathbf{r} = (r_1, \ldots, r_h)$ with $r_j \geq 0$ for $j \neq i$ and $r_i \geq 1$. Let $F_i(\mathbf{r}, k) = 0$ if $\sum_i r_i > k$ or $r_j < 0$ for some j or $r_i \leq 0$. Note as

$$\frac{E(\mathbf{m}+\mathbf{e}_{i+1}-\mathbf{e}_i, n)}{E(\mathbf{m}, n)} = \frac{E(\mathbf{m}+\mathbf{e}_{i+1}-\mathbf{e}_i, n)}{E(\mathbf{m}-\mathbf{e}_i, n)} \frac{E(\mathbf{m}-\mathbf{e}_i, n)}{E(\mathbf{m}, n)},$$

the miss probability $M(\mathbf{m}, v)$ equals

$$M(\mathbf{m}, v) = F_1(\mathbf{m}+\mathbf{e}_1, n) + \sum_{i=1}^{v} m_i \frac{F_{i+1}(\mathbf{m}+\mathbf{e}_{i+1}-\mathbf{e}_i, n)}{F_i(\mathbf{m}, n)}.$$

The next theorem establishes a recursive relationship for the $F_i(\mathbf{r}, k)$ values that allows us to compute the miss probabilities without causing underflows.

Theorem 2 *The miss probability, given by*

$$M(\mathbf{m}, v) = F_1(\mathbf{m}+\mathbf{e}_1, n) + \sum_{i=1}^{v} m_i \frac{F_{i+1}(\mathbf{m}+\mathbf{e}_{i+1}-\mathbf{e}_i, n)}{F_i(\mathbf{m}, n)},$$

and the quantities $F_i(\mathbf{r}, k)$, for $i = 1, \ldots, h$, $\mathbf{r} = (r_1, \ldots, r_h)$ with $r_j \geq 0$ for $j \neq i$ and $r_i \geq 1$, obey the following recursion

$$F_i(\mathbf{r}, k) = \frac{F_i(\mathbf{r}, k-1) + \sum_{j=1, j\neq i}^{h} p_k^j r_j \frac{F_i(\mathbf{r}-\mathbf{e}_j, k-1)}{F_j(\mathbf{r}-\mathbf{e}_i, k-1)} + p_k^i r_i}{1 + \sum_{j=1, j\neq i}^{h} \frac{p_k^j r_j}{F_j(\mathbf{r}-\mathbf{e}_i, k-1)} + \frac{p_k^i (r_i-1)\mathbf{1}_{\{r_i>1\}}}{F_i(\mathbf{r}-\mathbf{e}_i, k-1)}}, \tag{6}$$

and $F_i(\mathbf{e}_i, 1) = p_1^i$.

Proof By applying (4) on the numerator and denominator of (5) we get that $F_i(\mathbf{r}, k)$ can be written[1] as

$$\frac{E(\mathbf{r}, k-1) + \sum_{j=1}^{h} p_k^j r_j E(\mathbf{r}-\mathbf{e}_j, k-1)}{E(\mathbf{r}-\mathbf{e}_i, k-1) + \sum_{j=1}^{h} p_k^j (r_j - \mathbf{1}_{\{i=j\}}) E(\mathbf{r}-\mathbf{e}_i-\mathbf{e}_j, k-1)}.$$

By dividing the numerator and denominator by $E(\mathbf{r}-\mathbf{e}_i, k-1)$ (which is well defined as $r_i > 0$ and differs from zero as $\sum_s r_s - 1 \leq k-1$), we find

$$F_i(\mathbf{r}, k) = \frac{F_i(\mathbf{r}, k-1) + \sum_{j=1, j\neq i}^{h} p_k^j r_j \frac{E(\mathbf{r}-\mathbf{e}_j, k-1)}{E(\mathbf{r}-\mathbf{e}_i, k-1)} + p_k^i r_i}{1 + \sum_{j=1, j\neq i}^{h} \frac{p_k^j r_j}{F_j(\mathbf{r}-\mathbf{e}_i, k-1)} + \frac{p_k^i (r_i-1)\mathbf{1}_{\{r_i>1\}}}{F_i(\mathbf{r}-\mathbf{e}_i, k-1)}}.$$

The result follows by noting that $E(\mathbf{r}-\mathbf{e}_j, k-1)/E(\mathbf{r}-\mathbf{e}_i, k-1)$ can be written as $F_i(\mathbf{r}-\mathbf{e}_j, k-1)/F_j(\mathbf{r}-\mathbf{e}_i, k-1)$ when $r_j > 0$. □

Denote $\mathcal{R}_{k,h}$ as the set $\{(r_1, \ldots, r_h) | r_i \in \{0, \ldots, k\}, 1 \leq \sum_{i=1}^{h} r_i \leq k\}$. Algorithm 1 indicates how to use the above theorem to compute the values of $F_i(\mathbf{r}, k)$ for $i = 1, \ldots, h$, $k \leq n$ and $\mathbf{r} \in \mathcal{R}_k$ in $O(nh^2|\mathcal{R}_{m+1,h}|)$ time with $|\mathcal{R}_{m+1,h}| = O(m^h/h!)$. In other words we can determine the set of miss probabilities $M(\mathbf{m}, v)$ for $\mathbf{m} \in \mathcal{V}_h^{\leq m}$ in $O(nh^2 m^h/h!)$ time, where n is the number

Input: The vectors $\mathbf{p}$ and $\mathbf{m}$
Output: $M(\mathbf{m}, v)$ for $\mathbf{m} \in \mathcal{V}_h^{\leq m}$
for $i, j = 1$ *to* h **do**
  $F_j(\mathbf{e}_i, 1) = \mathbf{1}_{\{i=j\}} p_1^i$;
end
for $k = 1$ *to* n **do**
  for $\mathbf{r} \in \mathcal{R}_{\min(k,m+1),h}$ **do**
    for $i = 1$ *to* h **do**
      compute $F_i(\mathbf{r}, k)$ via (6);
    end
  end
end
for $\mathbf{m} = (m_1, \ldots, m_k) \in \mathcal{V}_h^{\leq m}$ **do**
  $M(\mathbf{m}, 0) = F_1(\mathbf{m} + \mathbf{e}_1, n)$;
  for $v = 1$ *to* $k - 1$ **do**
    $M(\mathbf{m}, v) = M(\mathbf{m}, v-1) + m_v \frac{F_{v+1}(\mathbf{m}+\mathbf{e}_{v+1}-\mathbf{e}_v, n)}{F_v(\mathbf{m}, n)}$;
  end
end

Algorithm 1: Overall miss probabilities $M(\mathbf{m}, v)$ for all $\mathbf{m} \in \mathcal{V}_h^{\leq m}$.

of items. Similarly, one can compute the set of miss probabilities $M(\mathbf{m}, v)$ for any $\mathbf{m} \in \mathcal{V}_h^{m_1,\ldots,m_h}$ in $O(nh^2 \prod_{i=1}^{h}(m_i + 1))$ time.

For the CLIMB algorithm we have $m_1 = \ldots = m_h = 1$ and $h{=}m$, meaning the miss probability can be computed in $O(nm^2 2^m)$ time, which is still exponential in m, but clearly a significant gain over directly relying on (2).

4.3 Per item miss probabilities

In this section we focus on the hit probability of item k, denoted as $H^{(k)}(\mathbf{m}, v)$, for $k = 1, \ldots, n$. We first show how to express $M^{(k)}(\mathbf{m}, v) = 1 - H^{(k)}(\mathbf{m}, v)$ in terms of $F_i(\mathbf{m}, n-1)$. Recall that the items $p_1, \ldots, p_n$ in the previous subsection were ordered in an arbitrary, but *fixed* order. To express $M^{(k)}(\mathbf{m}, v)$ in terms of $F_i(\mathbf{m}, n-1)$, we order the items such that item k, with request probability p_k, is the *last* item. Thus, we need to recompute the $F_i(\mathbf{m}, n-1)$ values for each item k and we denote these values as $F_i^{(k)}(\mathbf{m}, n-1)$. A direct application of this result would therefore leads to an algorithm with a time complexity $O(n^2)$. We further indicate how to reduce this to an algorithm in $O(n^{3/2})$.

Theorem 3 *The item k miss probability $M^{(k)}(\mathbf{m}, v)$ is*

$$M^{(k)}(\mathbf{m}, v) = \frac{1 + \sum_{i=1}^{v} \frac{p_k^i m_i}{F_i^{(k)}(\mathbf{m}, n-1)}}{1 + \sum_{i=1}^{h} \frac{p_k^i m_i}{F_i^{(k)}(\mathbf{m}, n-1)}},$$

if $\sum_{i=v+1}^{h} m_i < n$ and $M^{(k)}(\mathbf{m}, v) = 0$ otherwise.

[1] We use the notation $\mathbf{1}_{\{i=j\}}$ equals 1 if $i = j$ and 0 otherwise.

Proof The hit probability of item k is identical to the steady state probability that item k is in some list $i > v$:

$$M^{(k)}(\mathbf{m}, v) = 1 - \sum_{i=v+1}^{h} \sum_{j=1}^{m_i} \sum_{\mathbf{c} \in \mathcal{C}_n(m), c(i,j)=k} \pi(\mathbf{c}).$$

Let $E^{(k)}(\mathbf{r}, k)$ be defined as $E(\mathbf{r}, k)$ with the requirement that the items are ordered such that item k is in the last position. By (1), (3) and (4), we find

$$\begin{aligned} M^{(k)}(\mathbf{m}, v) &= 1 - \frac{\sum_{i=v+1}^{h} p_k^i m_i E^{(k)}(\mathbf{m} - e_i, n-1)}{E(\mathbf{m}, n)} \\ &= \frac{E^{(k)}(\mathbf{m}, n-1) + \sum_{i=1}^{v} p_k^i m_i E^{(k)}(\mathbf{m} - e_i, n-1)}{E^{(k)}(\mathbf{m}, n-1) + \sum_{i=1}^{h} p_k^i m_i E^{(k)}(\mathbf{m} - e_i, n-1)}. \end{aligned} \tag{7}$$

If $\sum_{i=v+1}^{h} m_i \geq n$, $E^{(k)}(\mathbf{m}, n-1)$ and $M^{(k)}(\mathbf{m}, v) = 0$ (as all items fit in the cache), otherwise $E^{(k)}(\mathbf{m}, n-1)$ differs from zero and the result follows from dividing the numerator and denominator by $E^{(k)}(\mathbf{m}, n-1)$. □

Note that p_k has to be the last item in the order when computing the $F_i^{(k)}(\mathbf{r}, n-1)$ values based on Theorem 2, thus for each k we must use a different order. However, it is also clear that if we only change the order of the last s items, the values of $F_i(\mathbf{r}, n-s)$ do not change.

To take advantage of this observation, we partition the set $\{1, \ldots, n\}$ of items into $\sqrt{n}$ sets $S_1, \ldots, S_{\sqrt{n}}$ each holding $\sqrt{n}$ items (for ease of presentation we assume that n is a square). To compute the hitting probabilities of the item k belonging to set S_j, we fix the order such that all the items *not* belonging to S_j appear first (in an arbitrary fixed order), followed by an arbitrary order of items of S_j with k as the last item. Thus, the computation of the $F_i(\mathbf{r}, n-\sqrt{n})$ values, with $r_i \in \{0, 1, \ldots, m\}$ and $\sum_i r_i \leq m$, is identical for all the items belonging to S_j and takes $O(nh^2m^h/h!)$ time, while computing the $F_i(\mathbf{m}, n-1)$ values from the $F_i(\mathbf{r}, n-\sqrt{n})$ values takes $O(\sqrt{n}h^2m^h/h!)$ time for each item in S_j. This results in an overall time complexity of

$$\sqrt{n} \cdot \left[O(\frac{nh^2m^h}{h!}) + \sqrt{n} \cdot O(\frac{\sqrt{n}h^2m^h}{h!}) \right] = O(\frac{n^{3/2}h^2m^h}{h!}),$$

which is a significant gain over $O(n^2h^2m^h/h!)$ if the number of items n is large.

For $h = 1$ we have $H^{(k)}(m, 0) = p_k m/(p_k m + F_1^{(k)}(m, n-1))$ and the above algorithm to compute the per item hit probabilities has a time complexity of $O(mn^{3/2})$. An efficient algorithm to compute the per item hit probabilities of the RANDOM and FIFO scheme appears to be novel as [10] only considered the overall hit probabilities.

Theorem 4 *For any* $\mathbf{m}$*, under the algorithms FIFO(*$\mathbf{m}, v$*) and RAND(*$\mathbf{m}, v$*),* $p_k > p_\ell$ *implies that* $M^{(k)}(\mathbf{m}, v) < M^{(\ell)}(\mathbf{m}, v)$*, that is, more popular items have lower miss probabilities.*

Proof The proof is a generalization of [37, Lemma 5] which established the same result for the CLIMB algorithm, that is, when $m_1 = \ldots = m_h = 1$ and $v = 0$. Due to (7), $M^{(k)}(\mathbf{m}) < M^{(\ell)}(\mathbf{m})$ is equivalent to

$$\sum_{i=v+1}^{h} p_\ell^i m_i E^{(\ell)}(\mathbf{m} - e_i, n-1) - \sum_{i=v+1}^{h} p_k^i m_i E^{(k)}(\mathbf{m} - e_i, n-1) < 0,$$

If we order the first $n-1$ entries in both cases such that items k and ℓ are in the last two positions and we use Equation (4), the above rewrites as

$$\begin{aligned}
&\sum_{i=v+1}^{h} p_\ell^i m_i \left(E(\mathbf{m} - e_i, n-2) + \sum_{j=1}^{h} p_k^j (m_j - \mathbf{1}_{\{i=j\}}) E(\mathbf{m} - e_i - e_j, n-2) \right) \\
&- \sum_{i=v+1}^{h} p_k^i m_i \left(E(\mathbf{m} - e_i, n-2) + \sum_{j=1}^{h} p_\ell^j (m_j - \mathbf{1}_{\{i=j\}}) E(\mathbf{m} - e_i - e_j, n-2) \right) \\
&= \sum_{i=v+1}^{h} (p_\ell^i - p_k^i) m_i E(\mathbf{m} - e_i, n-2),
\end{aligned}$$

which is negative as $p_k > p_\ell$. □

4.4 Upper and lower bounds on the overall miss probability

In the previous subsections we developed fast algorithms to compute the overall and per item miss probabilities of the FIFO$(\mathbf{m}, v)$ and RAND$(\mathbf{m}, v)$ algorithm. In this section we present a lower and upper bound on the overall miss probability when $v = 0$. Compared to Algorithm 1, that leads to an algorithm that is exponential in h, these bounds can be computed in a $O(mn)$ time (linear in the number of items times the cache size). The upper bound coincides with the miss probability of the original FIFO and RANDOM schemes. This means that using two or more lists always decrease the steady-state miss probability compared to using only one list (*i.e.*, RANDOM or FIFO).

Theorem 5 *The miss ratio $M(\mathbf{m}, 0)$ of the FIFO$(\mathbf{m}, 0)$ and RAND$(\mathbf{m}, 0)$ algorithm is upper bounded by*

$$M(\mathbf{m}, 0) \leq F_1((m+1)\mathbf{e}_1, n) = M_{\text{FIFO}}(\mathbf{m}),$$

and lower bounded by

$$F_1(\mathbf{e}_1 + m\mathbf{e}_h, n) = \frac{\sum_{\mathbf{x} \in C_n(m)} \left(\prod_{j=1}^{m} p_{x_j}^h \right) \left(1 - \sum_{j=1}^{m} p_{x_j}\right)}{\sum_{\mathbf{x} \in C_n(m)} \left(\prod_{j=1}^{m} p_{x_j}^h \right)}$$

with $m = \sum_{i=1}^{h} m_i$.

These bounds can be computed in $O(mn)$ time.

To establish the lower and upper bounds in Theorem 5 we rely on a Lemma of [34] based on the FKG inequality [12].

We order the items such that $p_1 \leq p_2 \leq \ldots \leq p_n$. For $\mathbf{x}, \mathbf{y} \in C_n(m)$ let $\mu(\mathbf{x}) = p_{x_1} p_{x_2} \ldots p_{x_m} > 0$ and define $\mathbf{x} \preceq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, \ldots, m$. A function f from $C_n(m)$ to $\mathbb{R}$ is increasing if $\mathbf{x} \preceq \mathbf{y}$ implies that $f(\mathbf{x}) \leq f(\mathbf{y})$, f is decreasing if $(-f)$ is increasing. Further, we state that f is permutation invariant if its value is independent of the order of its arguments $x_1, \ldots, x_m$.

Lemma 1 (Lemma of [34]) *Let g be decreasing and permutation invariant on $C_n(m)$. If f is increasing on $C_n(m)$, then*

$$\sum_{\mathbf{x} \in C_n(m)} \mu(\mathbf{x}) f(\mathbf{x}) g(\mathbf{x}) \sum_{\mathbf{y} \in C_n(m)} \mu(\mathbf{y}) \leq \sum_{\mathbf{x} \in C_n(m)} \mu(\mathbf{x}) f(\mathbf{x}) \sum_{\mathbf{y} \in C_n(m)} \mu(\mathbf{y}) g(\mathbf{y}).$$

If f is decreasing on $C_n(m)$, then

$$\sum_{\mathbf{x} \in C_n(m)} \mu(\mathbf{x}) f(\mathbf{x}) g(\mathbf{x}) \sum_{\mathbf{y} \in C_n(m)} \mu(\mathbf{y}) \geq \sum_{\mathbf{x} \in C_n(m)} \mu(\mathbf{x}) f(\mathbf{x}) \sum_{\mathbf{y} \in C_n(m)} \mu(\mathbf{y}) g(\mathbf{y}).$$

Proof (of Theorem 5) We start with the upper bound. Define the function g on $C_n(\mathbf{m})$ as

$$g(\mathbf{x}) = 1 - p_{x_1} - p_{x_2} - \ldots - p_{x_m}.$$

The function g is clearly decreasing and permutation invariant on $C_n(\mathbf{m})$. Let $f(\mathbf{x})$ be defined as

$$f(\mathbf{x}) = \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{x(i,j)} \right)^{i-1}.$$

where $x(i,j) = x_{\sum_{s=1}^{i-1} m_s + j}$. Due to (1) and (2) we have

$$\sum_{\mathbf{x} \in C_n(\mathbf{m})} \mu(\mathbf{x}) f(\mathbf{x}) g(\mathbf{x}) = Z(\mathbf{m}) M(\mathbf{m}),$$

$$\sum_{\mathbf{x} \in C_n(\mathbf{m})} \mu(\mathbf{x}) f(\mathbf{x}) = Z(\mathbf{m}),$$

while

$$\sum_{\mathbf{y} \in C_n(\mathbf{m})} \mu(\mathbf{y}) g(\mathbf{y}) = Z(\mathbf{m}) M_{FIFO}(\mathbf{m}) = Z(\mathbf{m}) F_1((\mathbf{m}+1)\mathbf{e}_1, n)$$

and

$$\sum_{\mathbf{y} \in C_n(\mathbf{m})} \mu(\mathbf{y}) = Z(\mathbf{m}).$$

As f is increasing, the upper bound therefore follows from the first inequality of Lemma 1.

For the lower bound define $\tilde{\mu}(\mathbf{x}) = \mu(\mathbf{x})^h \geq 0$ and let $\mathbf{x} \wedge \mathbf{y} = (\min(x_1, y_1), \ldots, \min(x_m, y_m))$ and $\mathbf{x} \vee \mathbf{y} = (\max(x_1, y_1), \ldots, \max(x_m, y_m))$, then

$\tilde{\mu}(\mathbf{x})\tilde{\mu}(\mathbf{y}) = \tilde{\mu}(\mathbf{x} \wedge \mathbf{y})\tilde{\mu}(\mathbf{x} \vee \mathbf{y})$ and $\tilde{\mu}$ is permutation invariant on $C_n(m)$. As a result, the proof of the Lemma in [34] can be repeated to show that Lemma 1 remains valid if μ is replaced by $\tilde{\mu}$. The result now follows from the second inequality of Lemma 1 (with μ replaced by $\tilde{\mu}$) by setting

$$f(\mathbf{x}) = \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{x(i,j)} \right)^{i-h},$$

which is a decreasing function on $C_n(m)$ and noting that

$$\frac{\sum_{\mathbf{y} \in C_n(\mathbf{m})} \tilde{\mu}(\mathbf{y}) g(\mathbf{y})}{\sum_{\mathbf{y} \in C_n(\mathbf{m})} \tilde{\mu}(\mathbf{y})} = F_1(\mathbf{e}_1 + \mathbf{m}e_h, n).$$

Theorem 2 implies that

$$F_h(r_h \mathbf{e}_h, k) = \frac{F_h(r_h \mathbf{e}_h, k-1) + p_k^h r_h}{1 + \frac{p_k^h (r_h - 1) \mathbf{1}_{\{r_h > 1\}}}{F_h((r_h - 1)\mathbf{e}_h, k-1)}},$$

and

$$F_1(\mathbf{e}_1 + r_h \mathbf{e}_h, k) = \frac{F_1(\mathbf{e}_1 + r_h \mathbf{e}_h, k-1) + p_k^h r_h \frac{F_1(\mathbf{e}_1 + (r_h - 1)\mathbf{e}_h, k-1)}{F_h(r_h \mathbf{e}_h, k-1)} + p_k}{1 + \frac{p_k^h r_h}{F_h(r_h \mathbf{e}_h, k-1)}}.$$

This shows that the quantities F_h and F_1 can be computed by induction on k and m. □

4.5 CLIMB is not optimal

Theorem 5 shows that the RANDOM algorithm achieves the worst performance (*i.e.*, the highest miss rate) for the class of RAND($\mathbf{m}, 0$) algorithms with $\sum_i m_i = m$ under the IRM model. Hence, it is tempting to conjecture that separating a list into two smaller lists improves the performance and that the CLIMB algorithm, that is, having m lists of size 1, achieves the lowest miss rate within this class. In [2, p135] an even stronger conjecture is presented that states that CLIMB is optimal under the IRM model for all finite-memory demand replacement algorithms.

Table 1 shows that both conjectures are false. The popularity distribution is $\mathbf{p} = (49, 49, 49, 49, 7, 1, 1)/205$ and the cache has a size $\sum_i m_i = 6$. Table 1 lists the ten vectors $\mathbf{m}$ with $\sum_i m_i = 6$ that achieve the lowest miss probability $M(\mathbf{m}, 0)$, together with their corresponding $M(\mathbf{m}, 0)$ value. For comparison, we also show the performance of RANDOM and LRU. This shows that CLIMB is not optimal. It further indicates that when fixing the number of lists h, the optimal choice for the length of each list is not necessarily setting $m_1 = \ldots = m_{h-1} = 1$: for instance, $\mathbf{m} = (1, 1, 3, 1)$ is better than $\mathbf{m} = (1, 1, 1, 3)$. This demonstrates that another of the *natural* conjectures formulated in [2, p135] is also false.

policy	$\mathbf{m}$	$M(\mathbf{m},0)$	lower bound
Optimal	RAND(1,1,4)	0.005284	0.004925
	RAND(1,1,3,1)	0.005299	0.004884
	RAND(1,1,2,2)	0.005317	0.004884
	RAND(1,1,2,1,1)	0.005321	0.004879
	RAND(1,1,1,3)	0.005338	0.004884
	RAND(1,1,1,2,1)	0.005343	0.004879
	RAND(1,1,1,1,2)	0.005347	0.004879
CLIMB	RAND(1,1,1,1,1,1)	0.005348	0.004878
	RAND(1,2,3)	0.005428	0.004925
	RAND(1,2,2,1)	0.005439	0.004884
LRU	LRU(6)	0.005880	–
RANDOM	RAND(6)	0.015350	0.015350

Table 1 CLIMB is not optimal for IRM model with $v = 0$: $p = (49, 49, 49, 49, 7, 1, 1)/205$ and $m = 6$.

4.6 Adding virtual lists does not always improve performance

As we will demonstrate in Section 6, separating a list into smaller lists or adding virtual lists that only store meta-data usually improve the hit rate under the IRM model. The previous example (Table 1) shows that separating a list into smaller lists does not always improve the performance. In Table 2, we also show that adding virtual lists can also decrease performance. We make use of the same popularity distribution $\mathbf{p} = (49, 49, 49, 49, 7, 1, 1)/205$ and consider a cache of size $\sum_i m_i = 6$. We compare the RANDOM policy (and CLIMB) with RANDOM plus one virtual list of size 1, one virtual list of size 2 and two virtual lists of size 1. In both cases, adding one virtual list decreases the miss probability but increasing the size of the first virtual list or adding a second virtual list increases the miss probability. Note that the increase of miss probability is very small and really depends on this specific example.

policy	$\mathbf{m}$	$M(\mathbf{m},v)$
RANDOM	RAND((4), $v = 0$),	0.14094006
	RAND((1,4), $v = 1$)	0.11139402
	RAND((2,4), $v = 1$)	0.12823856
	RAND((1,1,4), $v = 2$)	0.11389801
CLIMB	RAND((1,1,1,1), $v = 0$),	0.08041107
	RAND((1,1,1,1,1), $v = 1$)	0.06924691
	RAND((2,1,1,1,1), $v = 1$)	0.07576347
	RAND((1,1,1,1,1,1), $v = 2$)	0.07063632

Table 2 Adding virtual lists does not always improve performance for the IRM model with $p = (49, 49, 49, 49, 7, 1, 1)/205)$ and $m = 4$.

4.7 Miss probability is not Schur concave

A related question is whether the miss probability is a Schur-concave[2] function of the popularity distribution $\mathbf{p} = (p_1, \ldots, p_n)$ for the class of algorithms RAND$(\mathbf{m}, v)$. If the miss probability were Schur-concave, it would imply that putting more weight on the most popular items decreases the miss probability. The authors of [35] proved that the miss probability of the RANDOM algorithm, *i.e.*, when $h = 1$ and $v = 0$, is Schur-concave, which is almost immediate from [27, p80].

It is shown in the more recent paper [36] that the miss probability of the CLIMB algorithm is not a Schur-concave function of $\mathbf{p}$ by providing a simple counter-example with $m = 3$. We make use of the same example to demonstrate that a counter-example also exist for RAND$(\mathbf{m}, 0)$ with $h = 2$. We consider the popularity distributions $\mathbf{p}_1 = (0.45, 0.45, 0.05, 0.05)$ and $\mathbf{p}_2 = (.75, .15, 0.05, 0.05)$ ($\mathbf{p}(x_1)$ is majorized by $\mathbf{p}(x_2)$). For p_1, the miss probability is $M(1, 2) = 0.05835$ while it is $M(1, 2) = 0.05994$ for p_2. This implies that $M(1, 2)$ is not a Schur-concave function of $\mathbf{p}$.

5 Mean field approximation

In this section we develop an ODE approximation for the RAND$(\mathbf{m}, v)$ policy. We show that this approximation becomes exact as the number of items and the cache size tends to infinity. This approximation allows us to study fast and accurately the transient behavior of the RAND$(\mathbf{m}, v)$ algorithm. We use it to compute the time to fill an empty cache in Section 6. It can also be used to obtain a fast approximation of the steady-state miss probability, when the cache size or the number of lists make Algorithm 1 too time consuming.

5.1 Derivation of the equations and intuition

At a given time step t, item k is either part of some list $i \in \{1, \ldots, h\}$ or is not in any of the h lists. If an item is not in any of the h lists, we say that it is part of list 0. For an item $k \in \{1 \ldots n\}$ and a list $i \in \{0 \ldots h\}$, we define the random variables $X_{k,i}(t)$, where $X_{k,i}(t)$ equals 1 if the item k is part of list i at time t and 0 otherwise. The probability that item k is in list i at time t is $\mathbf{E}[X_{k,i}(t)]$.

Our mean-field approximation of the RAND$(\mathbf{m}, v)$ algorithm boils down to assuming that the evolution of two items is independent of each other. We approximate $\mathbf{E}[X_{k,i}(t)]$ by a deterministic quantity $x_{k,i}(t)$. The initial conditions of the ODE are $x_{k,i}(0) = 1$ if the item k is in the ith list of the

[2] A function is Schur-concave if $\mathbf{p}$ majorizes $\mathbf{p}'$, implies that the miss probability is lower for $\mathbf{p}$ than for $\mathbf{p}'$.

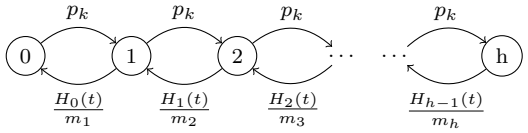

Fig. 1 Evolution of the list in which item k is. When p_k is small and the m_is are large, the state of one item becomes independent of the hit rate in each box and its behavior can be approximated by a time-inhomogeneous continuous-time Markov chain. This is the mean-field approximation.

cache at time $t = 0$ and 0 otherwise. Let $(x_{k,i}(t))_{k,i}$ be the unique[3] solution of the following set of ODEs, for $k \in \{1 \dots n\}, i \in \{1 \dots h\}$:

$$\dot{x}_{k,i}(t) = p_k x_{k,i-1}(t) - \sum_j p_j x_{j,i-1}(t) \frac{x_{k,i}(t)}{m_i} + \mathbf{1}_{\{i<h\}} \left(\sum_j p_j x_{j,i}(t) \frac{x_{k,i+1}(t)}{m_{i+1}} - p_k x_{k,i}(t) \right), \tag{8}$$

where $\mathbf{1}_{\{i<h\}}$ is equal to 1 if $i < h$ and 0 otherwise.

This equation can be understood as follows. Assume that item k is in list $i \in \{0 \dots h-1\}$ at time t. With probability p_k, item k is requested and moves to list $i+1$. With probability $H_{i-1}(t) = \sum_j p_j X_{j,i-1}(t)$, an item from list $i-1$ is requested and is exchanged with an item picked at random from list i. This item is item k with probability $1/m_i$. Hence, with probability $H_{i-1}(t)/m_i$, item k moves to list $i-1$. If the list in which item k is and the variables $H_i(t)$ were independent, the behavior of item k would be described by a Markov chain whose transition matrix is represented in Figure 1.

If the probability p_k is small and the list sizes m_i, for $i = 1, \dots, h$, are large, this Markov chain is well approximated by a continuous time Markov chain. If all items evolve independently, this leads to our mean-field approximation given by Equation (8).

In the next section, we establish a bound – Theorem 6 – that guarantees that, as the cache size grows, the transient hit probability can indeed be described by the ODE (8). One originality of our result is that our bound is valid regardless of the probability distribution (p_i). A more classical approach to obtain a convergence result would be to assume that items can be clustered into a finite number of classes and to let the number of items per class go to infinity. One could then use classical mean-field results [4,24,6]. This is the approach taken in [33]. Our approach is to relax the assumption of a finite number of classes with an infinite number of item per class. The drawback of our approach is that it requires a new proof. Yet, we believe that the cluster assumption is not a natural assumption for many mean-field models. Our approach shows that it is possible to obtain convergence results without assuming

[3] It should be clear that this ODE is Lipschitz-continuous and hence has a unique solution. Moreover, with the initial conditions indicated, the solution of the ODE satisfy that, for all time t: $\sum_{k=1}^{n} x_{k,i}(t) = m_i$ and $\sum_{i=0}^{h} x_{k,i}(t) = 1$.

that clusters exist, which makes the results more widely applicable. It could be generalized to other mean-field models for which the cluster assumption is not realistic.

5.2 Proof of the validity of the approximation

Let $H_i(t) = \sum_{k=1}^{n} p_k X_{k,i}(t)$ be the sum of the popularities of the items that are part of the ith list of the cache at time t and $H_0(t) = 1 - \sum_{i=1}^{h} H_i(t)$ be the miss probability at time t when $v = 0$. Let $\delta_i(t) = \sum_k p_k x_{k,i}(t)$, where $x_{k,i}(t)$ is the unique solution of ODE (8), with initial conditions $x_{k,i}(0) = X_{k,i}(0)$.

Theorem 6 *For any $T > 0$, there exists a constant $C > 0$ that depends on T such that, for any probability distribution over n items and list sizes $m_1 \dots m_h$, we have:*

$$\mathbf{E}\left[\sup_{t\in\{0\dots\tau\},i\in\{0\dots h\}} |H_i(t) - \delta_i(t)|\right] \leq C\sqrt{\max_{1\leq k\leq 1} p_k + \max_{0\leq i\leq h} \frac{1}{m_i}},$$

where $\tau := \lceil T/(\max_{k=1}^{n} p_k + \max_{i=0}^{h} \frac{1}{m_i})\rceil$.

In particular, Theorem 6 implies that the ODE approximation becomes exact as the cache size goes to infinity and the probability of each item goes to 0. This results can be seen as a generalization of [33] for two reasons. First, [33] only considers the RANDOM policy, that is, RAND($\mathbf{m}, v$) with $h = 1$. More importantly, [33] assumed that there were a finite number of classes of items with an infinite number of items per class. Our results is more general as we assume any distribution for $\mathbf{p}$.

In fact, we believe that this result can also be extended to show that the ODE approximation becomes exact as the caches size goes to infinity, even if the probability of some items do not go to 0 (as it is the case for a Zipf distribution with $\alpha > 1$). Assume that item k has a probability $p_k \gg 1/m_h$. In such a case, as shown of Figure 1, the item k will quickly enter list h by having been hit multiple times, but will only exit list h with probability $O(1/m_h) \ll p_k$. We believe a result similar to Theorem 6 can therefore be obtained by separating the behavior of the popular and non-popular items.

Proof Let $a = \max_k p_k$ and $b = \max_i 1/m_i$. Recall that $X_{k,i}(t)$ is equal to 1 if item k is in the ith list of the cache at time t and 0 otherwise. For any $\alpha \in \mathbf{Z}^+$, and $i \in \{0 \dots h\}$, we define $H_{i,\alpha}(t) = a^{1-\alpha} \sum_k (p_k)^\alpha X_{k,i}(t)$.

Let $\mathcal{H}$ be the set of infinite vectors $(\delta_{i,\alpha})_{i\in\{0\dots h\},\alpha\in\{1,2\dots\}}$ where $\delta_{i,\alpha} = a^{1-\alpha} \sum_k (p_k)^\alpha x_{k,i}$ for some $(x_{k,i})_{k,i}$ such that $\sup_{i,k} |x_{k,i}| < \infty$. We equip $\mathcal{H}$ with the ℓ_∞ norm: $\|h\|_\infty = \sup_{i,\alpha} |h|$ and define $\mathcal{H}_{\leq 1}$ the set of $h \in \mathcal{H}$ such that $\|h\|_\infty \leq 1$.

We define the function $f : \mathcal{H} \to \mathcal{H}$ by, for all $\delta \in \mathcal{H}$:

$$f_{i,\alpha}(\delta) = a\delta_{i-1,\alpha+1}(t) - \frac{\delta_{i-1,1}(t)\delta_{i,\alpha}(t)}{m_i} + \mathbf{1}_{\{i<h\}}\left(\frac{\delta_{i,1}(t)\delta_{i+1,\alpha}(t)}{m_{i+1}} - a\delta_{i,\alpha+1}(t)\right), \tag{9}$$

where $\mathbf{1}_{\{i<h\}}$ equals 1 if $i < h$ and 0 otherwise.

To prove the result we rely on the following lemma, the proof of which is postponed to Appendix A.

Lemma 2 *Let $(\mathcal{F}_t)$ be the filtration associated with process of the $(X_{k,i}(t))_{k,i}$, then:*

(i) $f(H(t))$ is the average variation of $H(t)$:

$$\mathbf{E}\left[H(t+1) - H(t)|\mathcal{F}_t\right] = f(H(t)) \tag{10}$$

(ii) the second moment is bounded:

$$\mathbf{E}\left[\|H(t+1) - H(t)\|_\infty^2 |\mathcal{F}_t\right] \leq 2a^2$$

(iii) there exists a constant L independent of the p_ks and the m_is such that the function f is Lipschitz-continuous of constant $L(a+b)$ on $\mathcal{H}_{\leq 1}$, where $a = \max_k p_k$ and $b = \max_i 1/m_i$.

(iv) If $x_{k,i}(t)$ is the unique solution of ODE (8), with initial conditions $x_{k,i}(0) = X_{k,i}(0)$, then the infinite vector δ, defined by $\delta_{i,\alpha}(t) = a^{1-\alpha}\sum_k (p_k)^\alpha x_{k,i}(t)$ is the unique solution of $\dot{\delta} = f(\delta)$, that is:

$$\delta(t) = H(0) + \int_0^t f(\delta(s))ds.$$

Let $\bar{H}$ denote the continuous function equal to $H(t)$ when $t \in \mathbf{Z}^+$ and linear between t and $t+1$. A straightforward computation shows that for all $t \in \mathbf{Z}^+$: $H(t) \in \mathcal{H}_{\leq 1}$ and that $\mathcal{H}_{\leq 1}$ is convex. Hence, for all $t > 0$: $\bar{H}(t) \in \mathcal{H}_{\leq 1}$.

Let $M(t) := \sum_{s=0}^{t-1} H(s+1) - H(s) - f(H(s))$. We have

$$\begin{aligned}\bar{H}(t) &= H(0) + \sum_{s=0}^{t-1} f(H(s)) + M(t) = H(0) + \int_{s=0}^{t} f(H(\lfloor s \rfloor))ds + M(t) \\ &= H(0) + \int_{s=0}^{t} f(\bar{H}(s))ds + M(t) + \int_0^t \left(f(H(\lfloor s \rfloor)) - f(\bar{H}(s))\right) ds.\end{aligned} \tag{11}$$

Equation (9) implies that for $h \in \mathcal{H}_{\leq 1}$, $\|f(h)\|_\infty \leq 2(a+b)$. As a result we get $\left\|H(\lfloor s \rfloor) - \bar{H}(s)\right\|_\infty \leq 2(a+b)(\lfloor s \rfloor - s)$, which by Lemma 2*(iii)* implies that

$$\begin{aligned}\int_{s=0}^{t} \left\|f(H(\lfloor s \rfloor)) - f(\bar{H}(s))\right\|_\infty ds &\leq tL(a+b)\int_0^1 2(a+b)sds \\ &= L(a+b)^2 t.\end{aligned}$$

Combined with (11) and Lemma 2*(iv)*, for $t < \tau$, this shows that $\left\|\bar{H}(t) - \delta(t)\right\|_\infty$ is less than

$$\int_0^t \left\|f(\bar{H}(s)) - f(\delta(s))\right\|_\infty ds + L(a+b)^2 t + \sup_{t \le \tau} \|M(t)\|_\infty ,$$

which, by Lemma 2*(iii)* and Grönwall's lemma implies that $\sup_{t\le\tau} \left\|\bar{H}(t) - \delta(t)\right\|_\infty$ is less than

$$\left(L(a+b)^2\tau + \sup_{t\le\tau} \|M(t)\|_\infty\right) \exp(L(a+b)\tau) \tag{12}$$

Moreover, by Lemma 2*(ii)*, we have $\mathbf{E}\left[\|M(\tau)\|_\infty^2\right] \le 2a^2\tau$. By Lemma 2*(i)*, $M(t)$ is a martingale. Therefore, this implies that $\mathbf{E}\left[\sup_{t\le\tau} \|M(t)\|_\infty\right] \le a\sqrt{2\tau} \le (a+b)\sqrt{2\tau}$. Replacing τ by $T/(a+b)$, Equation (12) is smaller than: $(LT(a+b) + \sqrt{a+b}\sqrt{2T}) \exp(LT)$. By construction, $a+b \le 2$, which implies that $a+b \le \sqrt{2}\sqrt{a+b}$. Thus, setting $C = \sqrt{2}(LT + \sqrt{T}) \exp(LT)$ gives the result. □

5.3 Steady-state behavior

Theorem 6 justifies the fact that the ODE is an approximation of the transient behavior of the hit probability of the original system. We now use this approximation to obtain a fixed-point equation for the steady-state hit probabilities. When $h = 1$ this equation is identical to the one introduced in [9] for FIFO and used in [28] for the RANDOM algorithm.

The next theorem shows that the mean field approximation of our system, given by the ODE (8) has a unique fixed point. As indicated by [4], having a unique fixed-point is not, in general, a sufficient condition to show that the steady-state of the stochastic system concentrates on this point. In our case due to Corollary 1 the stochastic process of RAND$(\mathbf{m}, v)$ is reversible and for reversible processes that converge to an ODE [6] showed that the stationary measure of a reversible process concentrates on the fixed point of the ODE. Although we cannot directly apply the result of [6] to our setting, we expect that a similar argument can be used. As RAND$(\mathbf{m}, v)$ and FIFO$(\mathbf{m}, v)$ have the same steady-state hit probability, this fixed-point provides a very efficient numerical method to compute the steady-state performance of both policies.

Theorem 7 *The mean-field model (8) has a unique fixed point. For this fixed point, the probability that item k is part of list i, for $k = 1, \dots, n$ and $i = 0, \dots, h$, is given by*

$$x_{k,i} = \frac{p_k^i z_i}{1 + \sum_{j=1}^h p_k^j z_j},$$

where $\mathbf{z} = (z_1, \dots, z_h)$ is the unique solution of the equation

$$\sum_{k=1}^n \frac{p_k^i z_i}{1 + \sum_{j=1}^h p_k^j z_j} = m_i. \tag{13}$$

Proof Let $(x_{k,i})_{k,i}$ be a fixed point of the ODE (8) and let $H_{i-1} = \sum_{j=1}^{n} p_j x_{j,i-1}$ be the corresponding hit probability in box $i-1$. The fixed-point equations are:

$$p_k x_{k,i-1} - \frac{H_{i-1}}{m_i} x_{k,i} + \mathbf{1}_{\{i<h\}} \big(\frac{H_i}{m_{i+1}} x_{k,i+1} - p_k x_{k,i} \big) = 0$$

As these equations correspond to the fixed-point equation of the birth-and-death process described in Figure 1, we have

$$x_{k,i} = \frac{p_k^i m_1 \dots m_i}{H_0 \dots H_{i-1}} x_{k,0}$$

Let $z_i = \prod_{j=0}^{i-1} m_{j+1}/H_j$ for $i \in \{1 \dots h\}$. By using that $\sum_{i=0}^{h} x_{k,i} = 1$, this implies that $x_{k,i} = p_k^i z_i / (1 + \sum_{j=1}^{h} p_k^j z_j)$. By using that $\sum_{k=1}^{n} x_{k,i} = m_i$, this yields

$$m_i = \sum_{k=1}^{n} x_{k,i} = \sum_{k=1}^{n} \frac{p_k^i z_i}{1 + \sum_{j=1}^{h} p_k^j z_j} \tag{14}$$

We now show that Equation (14) has a unique solution. For a vector $\mathbf{z} = (z_1 \dots z_h) \in (\mathbf{R}^+)^h$, we define $D_i(\mathbf{z})$, for $i = 1, \dots, h$, by

$$D_i(\mathbf{z}) = \sum_{k=1}^{n} \frac{p_k^i z_i}{1 + \sum_{j=1}^{h} p_k^j z_j} = \sum_{k=1}^{n} \frac{p_k^i}{1/z_i + \sum_{j=1}^{h} p_k^j z_j / z_i}$$

The function $D_i(\mathbf{z})$ is increasing in z_i and decreasing in z_j for $j \neq i$.

For a vector $\mathbf{z} = (z_1, \dots, z_h)$, $i \in \{1, \dots, h\}$ and $y > 0$, we denote $\mathbf{z}_{-i}(y)$ the vector whose coordinates are all equal to the ones of $\mathbf{z}$ except for the ith one which is equal to y. $D_i(\mathbf{z}_{-i}(y))$ is increasing in y. Hence, the equation $D_i(\mathbf{z}_{-i}(y)) = m_i$ has a unique solution that we denote $G_i(\mathbf{z})$ (it should be clear that, by definition, $G_i(\mathbf{z})$ does not depend on z_i). Moreover, as $D_j(\mathbf{z})$ is decreasing in z_j for $j \neq i$, $G_i(\mathbf{z})$ is increasing in z_j, for all $j \neq i$. This shows that G is increasing (componentwise) in $\mathbf{z}$: if for all i, $z_i \leq z_i'$, then for all i: $G_i(\mathbf{z}) \leq G_i(\mathbf{z}')$.

We define the sequence $\mathbf{z}^t$ by $\mathbf{z}^0 = (0, \dots, 0)$ and $\mathbf{z}^{t+1} = G(\mathbf{z}^t)$ for $t \geq 0$. As $G(\mathbf{z})$ is componentwise increasing in $\mathbf{z}$, the sequence $\mathbf{z}^t$ is increasing. Moreover, by using that $\mathbf{z} = \mathbf{z}^t_{-i}(z_i^t)$ and $m_i = D_i(\mathbf{z}^t_{-i}(z_i^{t+1}))$, we have:

$$D_i(\mathbf{z}^t) = D_i(\mathbf{z}^t_{-i}(z_i^t)) \leq D_i(\mathbf{z}^t_{-i}(z_i^{t+1})) = m_i$$

where the inequality from the fact that z_i^t is increasing in t and $D_i(\mathbf{z})$ is increasing in z_i. This shows for all t:

$$n - \sum_{i=1}^{h} m_i \leq n - \sum_{i=1}^{h} D_i(\mathbf{z}^t) = \sum_{k=1}^{n} \frac{1}{1 + \sum_{j=1}^{h} p_k^j z_j^t},$$

which implies that the sequence $\mathbf{z}^t$ is bounded. As this sequence is increasing, it converges to a fixed point of G.

α	n	m_1	m_2	exact	mean field
0.8	300	2	98	0.3466	0.3470
		30	70	0.3608	0.3612
		98	2	0.4239	0.4245
0.8	3000	20	980	0.3034	0.3035
		300	700	0.3159	0.3160
		980	20	0.3723	0.3724
1.1	300	2	98	0.1719	0.1722
		30	70	0.1832	0.1835
		98	2	0.2362	0.2367
1.1	3000	20	980	0.1110	0.1110
		300	700	0.1183	0.1183
		980	20	0.1531	0.1531

Table 3 Mean field model validation of the long run miss probability for $v = 0$ and $h = 2$ with Zipf-like popularity distributions.

We now prove the uniqueness. Let $\lambda > 1$. If we multiply all coordinates of $\mathbf{z}$ by λ, the $D(\lambda\mathbf{z})$ becomes:

$$D_i(\lambda\mathbf{z}) = \sum_{k=1}^{n} \frac{p_k^i z_i \lambda}{(1 + \sum_{j=1}^{h} p_k^j z_j \lambda)} = \sum_{k=1}^{n} \frac{p_k^i z_i}{(1/\lambda + \sum_{j=1}^{h} p_k^j z_j)} > D_i(\mathbf{z}).$$

This implies that $G_i(\lambda z) < \lambda G_i(\mathbf{z})$.

The function G satisfies the assumptions of a standard interference function I of [38], which means that it is positive, monotone and sub-homogeneous. By [38, Theorem 2], this implies that the fixed point of G is unique and that the iterations $\mathbf{z}^{t+1} = G(\mathbf{z}^t)$ converge, regardless of $\mathbf{z}^0$. This results can be proven by showing that $x \mapsto G(\exp(x))$ is a contraction. □

5.4 Numerical algorithm and validation

The proof of Theorem 7 is based on the construction of an iterative scheme $\mathbf{z}^{t+1} = G(\mathbf{z}^t)$ that converges to the unique fixed point given by Equation (13). This iteration provides an efficient method to compute the fixed-point, as it converges exponentially fast to the fixed point and each iteration takes $O(nh)$ time, that is, is linear in the number of items times the number of lists. This is clearly a significant improvement over the $O(nhm^h/h!)$ time complexity of Algorithm 1, used to compute exact hit probabilities, especially for large cache size and/or a large number of lists.

In Tables 3 and 4, we compare the steady-state miss probabilities given by the mean-field approximation with the exact values given by Algorithm 1 when $v = 0$. The popularity distribution follows a Zipf-like distribution of parameter 0.8 or 1.1. These tables contain the results for various values of $\mathbf{m}$. We observe that, in all cases, the approximation is within 1% of the exact value. This holds even when the lists are small or with highly non-uniform distributions (Zipf with $\alpha > 1$).

m_1	m_2	m_3	m_4	exact	mean field
2	2	96	–	0.3166	0.3169
10	30	60	–	0.3296	0.3299
20	2	78	–	0.3273	0.3276
90	8	2	–	0.4094	0.4100
1	4	10	85	0.3039	0.3041
5	15	25	55	0.3136	0.3139
25	25	25	25	0.3345	0.3348
60	2	2	36	0.3514	0.3517

Table 4 Mean field model validation of the long run miss probability for $v = 0$, $h = 3$ and $h = 4$ with $n = 300$ and a Zipf-like popularity distribution with $\alpha = 0.8$.

α	m_i	v	simul	mean field
0.5	30	0	.50113 ± .00011	.50116
		3	.57850 ± .00008	.57848
0.75	$10 + 40 \cdot \mathbf{1}_{\{i>5\}}$	0	.32307 ± .00002	.32310
		6	.41049 ± .00005	.41053
0.8	$10i$	0	.15836 ± .00003	.15838
		1	.16209 ± .00003	.16212
0.9	$30 - (i-5)^2$	0	.29437 ± .00003	.29439
		2	.31541 ± .00003	.31546
1.1	$8(11-i)$	0	.09412 ± .00007	.09417
		7	.35301 ± .00008	.35351
1.4	$8 + 72 \cdot \mathbf{1}_{\{i \text{ is odd}\}}$	0	.02504 ± .00001	.02504
		4	.04057 ± .00001	.04057

Table 5 Mean field model validation of the long run miss probability for $h = 10$ lists with $n = 1000$ and a Zipf-like popularity distribution based on 5 simulation runs.

When the number of lists exceeds five, the execution time of Algorithm 1 becomes prohibitive (it grows as m^h). We therefore compare the mean-field model with values obtained by simulation for $h = 10$ and $v \geq 0$ in Table 5. We show the results for various popularity models (Zipf 0.5 to Zipf 1.4) and diverse lists sizes: all lists have the same size ($m_i = 30$), increasing sizes ($m_i = 10i$ or $\mathbf{m} = (10, \ldots, 10, 50, \ldots, 50)$), varying ($m_i = 30 - (i-5)^2$), decreasing sizes ($m_i 8(11-i)$) or alternating ($\mathbf{m} = (80, 8, 80, 8 \ldots)$). In all cases, the mean-field approximation was computed almost instantaneously on a regular laptop while the time to obtain our simulation results was several hours. The accuracy of the transient behavior of the ODE approximation is discussed in Section 6.1.

6 Numerical results and guidelines

In this section, we wish to formulate some guidelines on how to select the number of lists and the list sizes m_1 to m_h. We first focus on the IRM model and study the trade-off between time to fill the cache and miss probability. We then explore traces of real data.

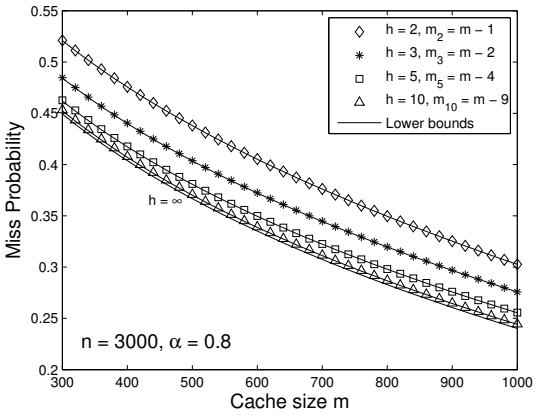

Fig. 2 Lower bounds for $h = 2, 3, 5, 10$ and ∞ and the miss rate $\mathrm{M}(\mathbf{m}, 0)$ with $m_1 = \ldots = m_{h-1} = 1$ for $h = 2, 3, 5$ and 10 as a function of the cache size m, with $v = 0$, $n = 3000$ and $\alpha = 0.8$.

6.1 IRM model and time to fill the cache

6.1.1 Influence of the number of lists

Figure 2 depicts the lower bounds established in Theorem 5 for various h values as a function of the cache size m when $n = 3000$ and $\mathbf{p}$ follows a Zipf-like distribution with $\alpha = 0.8$, that is, $p_i = A/i^{\alpha}$, where A is the normalizing constant. Furthermore, we also plotted the miss probability $\mathrm{M}(\mathbf{m}, 0)$ of the $\mathrm{RAND}(\mathbf{m}, 0)$ algorithm when $m_1 = \ldots = m_{h-1} = 1$. The main conclusions are first that having more lists generally improves the performance[4]. Second, these lower bounds appear to be very sharp under a Zipf-like workload (with $\alpha = 0.8$ and $n = 3000$ items). Third, the figure demonstrates that there is little room for further reducing the miss probability by using more than $h = 10$ lists. Most of the gain can be obtained by implementing a limited number of lists. This observation will be confirmed further on using trace-based simulations.

In Figure 3 we study the impact of the size m_1 of the first list on the miss probability when $h = 2$, $m = 300$, $n = 1000$ and the popularity distribution $\mathbf{p}$ follows a Zipf-like distribution with α ranging from 0.5 to 1. As reported in [7], many traces corresponds to Zipf-like distribution with parameters $\alpha \in [.6, .9]$. Our range of values α correspond to a super-set of these values. The figure indicates that there is no need to make the size m_1 of the first list extremely small in order to have a miss probability that is close to the lower bound. This is important as we will show next that small m_1 values make it harder for items that suddenly become popular to enter the cache.

[4] Although adding too many lists can be detrimental for the performance for some very specific and skewed popularity distribution (as shown in Section 4.5).

6.1.2 Influence of the first list size

To this end, we study the transient behavior of the hit probability under the IRM model starting from an empty cache by making use of the mean field model (with $H_0(t) = 1$). We also validate the accuracy of the mean field model to capture the transient regime by plotting the corresponding simulation results (averaged over 5 or 25 runs to reduce the noise). Figure 4 shows the transient behavior of the hit probability as a function of the number of requests received under the IRM model with a Zipf-like popularity distribution with parameters[5] $\alpha = 0.5$ for $h = 2$ lists, $m = 200$, $n = 1000$ and various values for the size of the first list m_1. The figure indicates that the mean field model and the simulation based results are in perfect agreement. Further, as expected, it demonstrates that while decreasing m_1 improves the steady state hit probability (which is in agreement with Figure 3), the time to reach the steady state increases as a function of m_1. Our conclusion is that the time to fill the cache depends mainly on the size of the first list. Assigning a sufficient portion of the overall cache size, to the first list gives a significant increase in the hit probability while having a limited impact on the cache reactivity.

To obtain more insights on the time to fill the cache, let us focus on the case when $\mathbf{p}$ is a uniform distribution. In this case, the mean-field approximation (8) simplifies to

$$\dot{x}_i(t) = \frac{x_{i-1}(t)}{n} - \frac{x_{i-1}(t)x_i(t)}{m_i} + \mathbf{1}_{\{i<h\}}\left(\frac{x_i(t)x_{i+1}(t)}{m_{i+1}} - \frac{x_i(t)}{n}\right).$$

When $m_1 \ll n$, items enter the first list at rate $1/n$ and leave it at rate $1/m_1$. This leads to $x_1(t) \approx (m_1/n)(1 - \exp(-t/m_1))$: the time to fill the first list is proportional to m_1. When list i contains $x \approx m_1$ items, these items jump to

[5] The choice of $\alpha = 0.5$ is arbitrary and similar results can be observed with larger α.

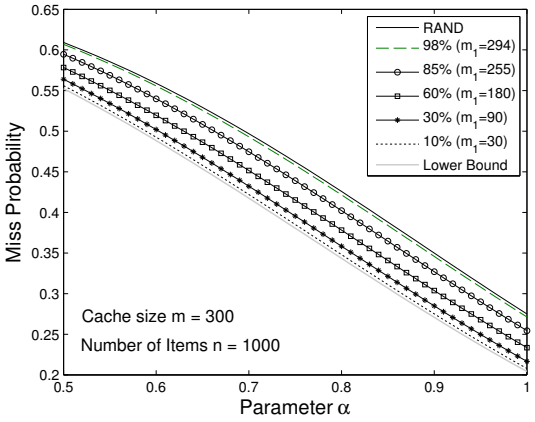

Fig. 3 Impact of parameter α of the Zipf-like distribution on the miss probability $M(m_1, m_2)$, with $v = 0$, $n = 1000$ and $m = 300$.

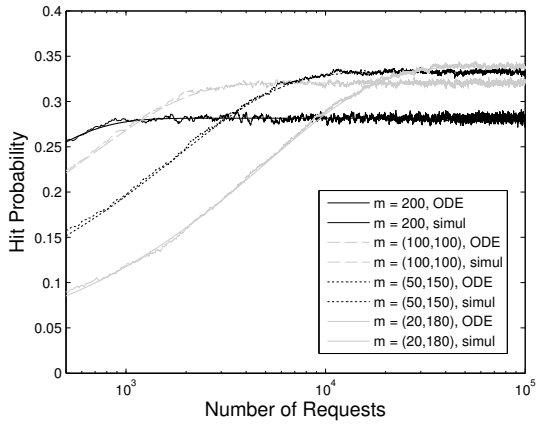

Fig. 4 Evolution of the hit probability starting from an empty cache with $v = 0$, $n = 1000$ and $\alpha = 0.5$. Simulation is based on 25 runs.

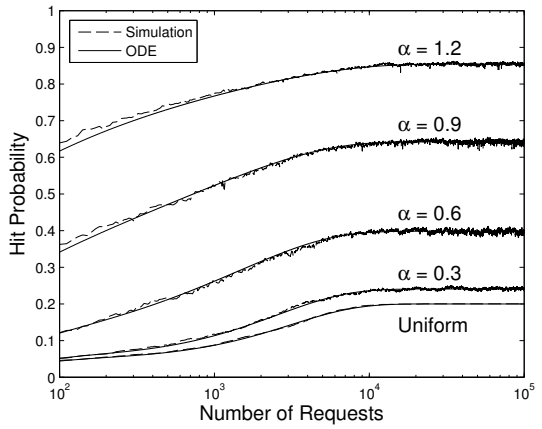

Fig. 5 Evolution of the hit probability starting from empty cache with $v = 0$, $n = 1000$ and $\mathbf{m} = (50, 150)$. Simulation is based on 5 runs.

the next list at rate $x/n \approx m_1/n$. This suggests that the time to fill the lists 2 to h is of the order $n(m_2/m_1 + m_3/m_2 + \cdots + m_h/m_{h-1})$. Hence, a natural choice is to set $m_1 = m_2 = \cdots = m_h$, which results in a total time to fill the cache of $m_1 + n(m - m_1)/m_1$.

Although this analysis only works for a uniform distribution, we show in Figure 5 that the impact of the parameter α of the Zipf-like popularity distribution $\mathbf{p}$ on time to reach the steady state hit probability is limited. This figure shows the hit probability as a function of the number of requests for $n = 1000$ items and $\mathbf{m} = (50, 150)$. Similarly, we depict the transient hit probability when $m_1 = 40$ for $h = 2, 3$ and 4 lists Figure 6. The main message of this figure is that the time to reach the steady state hit probability does not depend much on the list sizes m_2 to m_h. Hence, instead of working with

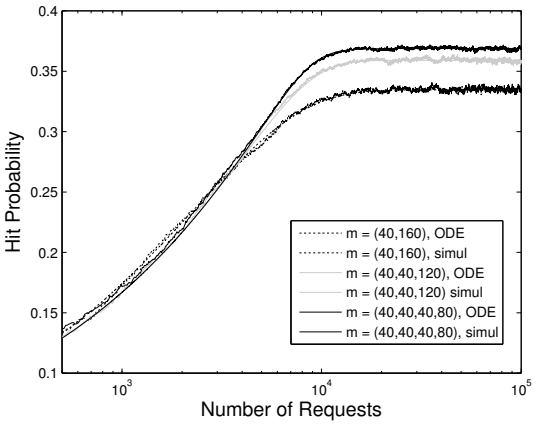

Fig. 6 Evolution of the hit probability starting from empty cache with $v = 0$, $n = 1000$ and $\alpha = 0.5$. Simulation is based on 25 runs.

just two lists, one can improve the steady state hit probability by relying on multiple lists without affecting the reactivity of the cache when the size of each of the lists is at least as large as the first one.

6.1.3 Influence of the presence of meta-data lists

Let us now study the impact of adding additional lists that only store meta data to a cache. For this purpose consider the case where $n = 1000$, $\alpha = 0.5$ and $m_i = 40$ for all i. Figure 7 demonstrates what happens to the time to fill the cache when $h = 5 + v$, for $v = 0, \ldots, 3$, meaning the cache can store 200 items in all cases and increasing v by one adds another list of size 50 that contains meta data only. This example demonstrates that adding extra lists with meta data to a cache typically improves the long run hit probability under the IRM model and the additional gain obtained from adding another list decreases as more lists are added. However increasing v also an increases in the time needed to fill the cache. As such it seems best not to add too many lists with meta data as this makes the cache replacement policy very inflexible when faced with dynamic workloads.

In the previous example all lists were of size 40. As storing only meta data requires less storage capacity than storing the actual items, one can easily make use of larger m_1 to m_v values at limited cost. Figure 8 considers the same example as before with $v = 1$ and studies the impact of m_1, the size of the list with meta data only. It illustrates that increasing m_1 typically increases the time to fill the cache while it reduces the long run hit probability (by a very small fraction for small m_1 values).

Our conclusion is that the addition of meta-data lists leads to better performance but slower reactivity. A good balance between the long run hit probability and the cache responsiveness therefore appears to exist in setting the

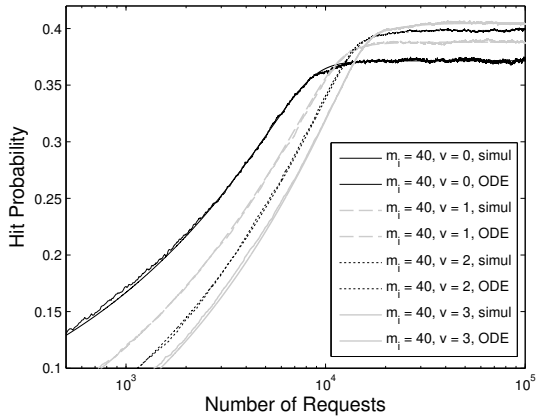

Fig. 7 Evolution of the hit probability starting from empty cache with $h = v + 5$, $n = 1000$ and $\alpha = 0.5$. Simulation is based on 25 runs.

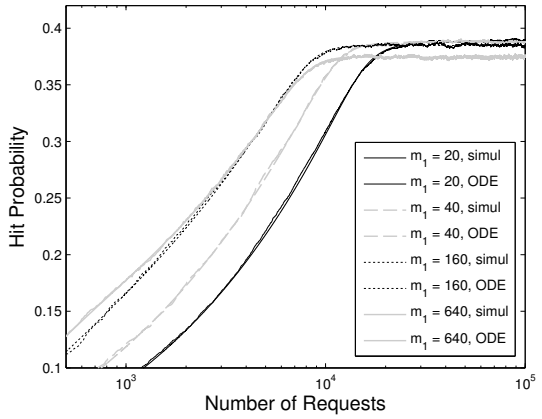

Fig. 8 Evolution of the hit probability starting from empty cache with $n = 1000$ and $\alpha = 0.5$. There is one meta-data list of size $m_1 \in \{20 \dots 640\}$. Simulation is based on 25 runs.

meta data lists sizes equal to the size of the lists that contain the items stored in the cache.

6.2 Trace-driven simulations

To validate the insights obtained by studying the IRM model, we perform trace-based simulations using a publicly available YouTube trace that was collected at a campus network during a 14 day period in 2008 [39, Trace T5]. This trace contains a total of 611,968 requests for 303,331 different videos, meaning that, if we start with an initially empty cache, the miss probability is at least $303,331/611,968 \approx 0.496$ even if the cache has infinite capacity.

Further, about 65.9% of these videos were requested only once during the trace, meaning the request pattern is quite different from the IRM model. We selected this particular trace among the ones discussed in [39] as it is by far the longest one.

We developed a simulation program for the FIFO($\mathbf{m}, 0$), RAND($\mathbf{m}, 0$), strict FIFO($\mathbf{m}, 0$) and LRU($\mathbf{m}, 0$) algorithms discussed in Section 3, for the scenario where the campus network relies on a single proxy cache. As only HTML-based control messages are available for trace T5, we have no information on the size of the requested videos. We therefore assumed (for all the policies) that the cache size is expressed in the number of videos m that can be stored in the proxy cache (the average size of a video should be a few Mbyte). This should not impact the results much if the correlation between video popularity and video size is small.

In all figures of this section, we only report results for $v = 0$. The reason is that, for this particular trace, the performance when $v \geq 1$ is worse. Indeed, for this particular trace, only 200k items were requested twice, 160k three times and 130k four times. This shows that by having a $v = 1, 2$ or 3, one cannot achieve a hit rate above $0.3354, 0.2580$ and 0.2122, as the first $v+1$ requests for any video cause a miss even if all the lists have infinite capacity. Just to state one number for $v = 1$: the LRU($\mathbf{m}, 1$) algorithm with $h = 6, v = 1, m_1 = \infty$ and $m_i = 1000$ for $i > 1$, achieves a hit rate of 0.2329 which is well below the hit rates reported further on for $v = 0$.

When comparing the different replacement policies, we varied the size of the first list m_1 and divided the remaining cache capacity equally among the remaining $h - 1$ lists. Figures 9 to 11 depict the hit probability as a function of the cache capacity assigned to list 2 to h when the total cache capacity is $m = 5000$ videos and we make use of $h = 2, 3$ or 5 lists. The following insights are provided by these figures. While the FIFO($\mathbf{m}, 0$) and RAND($\mathbf{m}, 0$) algorithm perform alike under the IRM model, RAND($\mathbf{m}, 0$) results in a lower hit probability when using real data, as expected. The strict FIFO($\mathbf{m}, 0$) and FIFO($\mathbf{m}, 0$) algorithm do however perform very similar, especially if enough capacity is assigned to list 2 to h. The LRU($\mathbf{m}, 0$) algorithm does outperform FIFO($\mathbf{m}, 0$) when using two or three lists, but the gain becomes very limited when using $h = 5$ lists. We also note that FIFO($\mathbf{m}, 0$) clearly outperforms a pure LRU cache.

In all cases, the hit probability is maximal when about 30 to 60% of the cache capacity is assigned to the first list. It quickly deteriorates as the size of the first list becomes small. This observation is in line with the earlier observation that the time to fill the cache is larger for small values of m_1, that make it harder to insert new items into the cache.

The increase in the hit probability as a function of the number of lists is the most pronounced when h is small, which is in agreement with earlier observations under IRM. In fact, further increasing the number of lists beyond 10 causes a decrease in the hit probability, mostly because the sizes of the lists 2 to h become small, making it harder to insert new items in the cache.

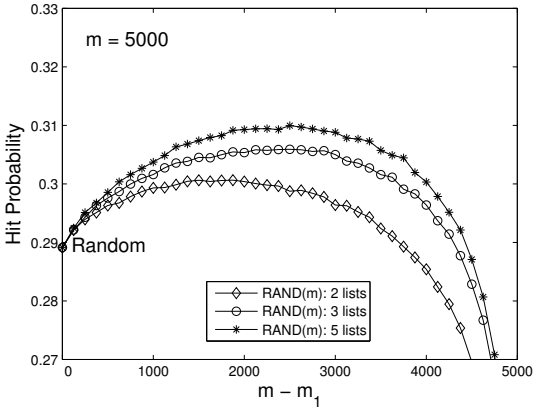

Fig. 9 Trace-based hit probability of RAND($\mathbf{m}, 0$) with $m_i = (m - m_1)/(h - 1)$ for $i = 2, \ldots, h$ and $\sum_i m_i = 5000$.

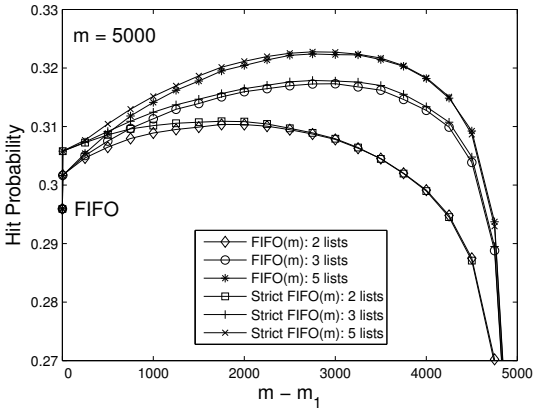

Fig. 10 Trace-based hit probability of FIFO($\mathbf{m}, 0$) and strict FIFO($\mathbf{m}, 0$) with $m_i = (m - m_1)/(h - 1)$ for $i = 2, \ldots, h$ and $\sum_i m_i = 5000$.

Both strict FIFO($\mathbf{m}, 0$) and FIFO($\mathbf{m}, 0$) exhibit a jump in the hit probability as soon as we add a second list, even if this list can only store a single item. For strict FIFO($\mathbf{m}, 0$) this can be easily understood as the second list allows an item that is hit repeatedly to move to the front of the cache. In particular, strict FIFO($m-1, 1, 0$) behaves very similar to a pure LRU, that is known to outperform pure FIFO [34]. The same happens to a lesser extent for FIFO($\mathbf{m}, 0$), but items now move to a random position in the first list when being demoted. Still, on average it allows popular items that are in the back of the cache to move away from the back of the cache. RAND($\mathbf{m}, 0$) does not exhibit such a behavior as being at the front or back in the first list makes no difference.

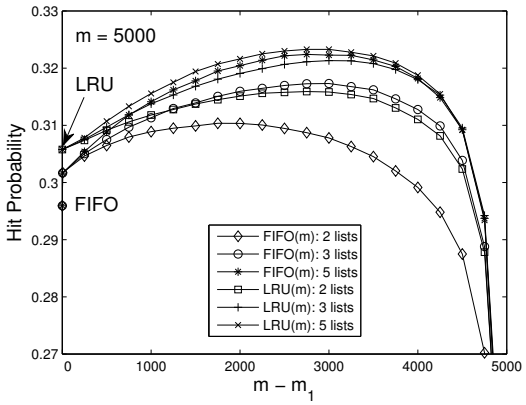

Fig. 11 Trace-based hit probability of FIFO($\mathbf{m}, 0$) and LRU($\mathbf{m}, 0$) with $m_i = (m - m_1)/(h-1)$ for $i = 2, \ldots, h$ and $\sum_i m_i = 5000$.

7 Conclusion

In this paper, we studied a family of cache replacement algorithms FIFO($\mathbf{m}, v$) and RAND($\mathbf{m}, v$). We provided close-form results for the steady-state probabilities under the IRM model, as well as a polynomial algorithm to compute the steady-state miss probabilities. We further developed a mean-field approximation that provides a fast and accurate method for approximating the miss probabilities. We used this approximation to study the transient behavior of the cache and to provide guidelines of how to tune the number of lists and the list sizes. This suggests that the first few lists should be large enough and have equal sizes. By using these insights, we verify on real traces of cache requests that these policies perform well and can outperform other classical heuristics such as FIFO or LRU. We also studied the used of virtual lists, suggested in [28] and that only store meta-data. We found that the number of virtual lists should be limited (one or two) and that these virutal lists should be similar in size to the lists containing the actual items. Our trace-based experiments also show that when many items are requested a small number of times, having virtual lists is detrimental.

A direct extension of this work would be to consider a network of caches, by using the model of [5,11]. We also aim at obtaining close-form expression for the steady-state probabilities or the time to fill the cache when the distribution is Zipf(α) or develop an approximation for studying the transient behavior of FIFO($\mathbf{m}, v$). Considering more general request models than IRM like MAP arrival or shot-noise is also important. For MAP arrivals, the steady-state results for LRU($\mathbf{m}, v$) obtained in [15] could be generalized to the case of RAND($\mathbf{m}, v$) and FIFO($\mathbf{m}, v$).

8 Acknowledgment

This work is partially supported by the EU project QUANTICOL, 600708.

A Proof of Lemma 2

Let $\alpha \in \mathbf{Z}^+$ and $i \in \{1 \dots h\}$. Two types of events can modify the value of $H_{i,\alpha}(t)$:

- If at time t, the item k is requested and is in list $i-1$, then, it is exchanged with another item j that is chosen uniformly at random from list i. This occurs with probability $X_{k,i-1}(t)X_{j,i}(t)p_k/m_i$ and modifies $H_{i,\alpha}(t)$ in $H_{i,\alpha}(t) + a^{1-\alpha}(p_k^\alpha - p_j^\alpha)$.
  The average variation of $H_{i,\alpha}$ due to these events is:

$$\begin{aligned}\sum_{k,j} X_{k,i-1}(t)X_{j,i}(t)a^{1-\alpha}(p_k^\alpha - p_j^\alpha)\frac{p_k}{m_i} &= \sum_k X_{k,i-1}(t)a^{1-\alpha}p_k^{\alpha+1}\sum_j X_{j,i}(t)\frac{1}{m_i} \\ &\quad + \sum_k X_{k,i-1}(t)p_k \sum_j a^{1-\alpha}p_j^\alpha X_{j,i}(t)\frac{1}{m_i} \\ &= aH_{i-1,\alpha+1}(t) + \frac{H_{i-1,1}(t)H_{i,\alpha}(t)}{m_i},\end{aligned}$$

  where the last line comes from the fact that $\sum_j X_{j,i}(t) = m_i$ and $\sum_k p_k X_{k,i-1}(t) = H_{i-1,1}(t)$.
- If $i < h$ and if at time t, the item k is in cache i and is hit, then, it is exchanged with another item j that is chosen uniformly at random from cache $i+1$. This occurs with probability $X_{k,i}(t)X_{j,i+1}(t)p_k/m_{i+1}$ and modify $H_{i,\alpha}(t)$ in $H_{i,\alpha}(t) + a^{1-\alpha}(-p_k^\alpha + p_j^\alpha)$. This second type of events leads to an average variation of $aH_{i,\alpha+1}(t)+(H_{i,1}(t)H_{i+1,\alpha}(t))/m_{i+1}$ if $i < h$.

Summing the two terms implies *(i)*: for all i, α, we have

$$\mathbf{E}\left[H_{i,\alpha}(t+1) - H_{i,\alpha}(t)|\mathcal{F}_t\right] = f_{i,\alpha}(H(t)).$$

We now show that the second moment of the variation of H is bounded. For all k, j $0 \le p_k, p_j \le a$, which implies $a^{2-2\alpha}(p_k^\alpha - p_j^\alpha)^2 \le a^2$. Therefore, by using the same two types of events as for the proof of *(i)*, we have

$$\begin{aligned}\mathbf{E}\left[\sup_\alpha (H_{i,\alpha}(t+1) - H_{i,\alpha}(t))^2|\mathcal{F}_t\right] &= \sum_{k,j} X_{k,i-1}(t)X_{j,i}(t)\sup_\alpha a^{2-2\alpha}(p_k^\alpha - p_j^\alpha)^2\frac{p_k}{m_i} \\ &\quad + \sum_{k,j} X_{k,i}(t)X_{j,i+1}(t)\sup_\alpha a^{2-2\alpha}(p_j^\alpha - p_k^\alpha)^2\frac{p_k}{m_{i+1}} \\ &\le \sum_{k,j} X_{k,i-1}(t)X_{j,i}(t)a^2\frac{p_k}{m_i} + \sum_{k,j} X_{k,i}(t)X_{j,i+1}(t)\frac{a^2 p_k}{m_{i+1}} \\ &= (H_{i-1,1}(t) + H_{i,1}(t))a^2\end{aligned}$$

This shows that

$$\begin{aligned}\mathbf{E}\left[\|H(t+1) - H(t)\|_\infty^2|\mathcal{F}_t\right] &= \mathbf{E}\left[\sup_{i,\alpha}(H_{i,\alpha}(t+1) - H_{i,\alpha}(t))^2|\mathcal{F}_t\right] \\ &\le \mathbf{E}\left[\sum_i \sup_\alpha (H_{i,\alpha}(t+1) - H_{i,\alpha}(t))^2|\mathcal{F}_t\right] \\ &\le \sum_i (H_{i-1,1}(t) + H_{i,1}(t))a^2 \le 2a^2.\end{aligned}$$

The points *(iii)* and *(iv)* are easier as f is a sum of two terms: the first one is Lipschitz-continuous of constant a and the second is a second order polynomial function of h divided by m_i, which is therefore Lipschitz-continuous of constant C/m_i on all bounded sub-space of $\mathcal{H}$. Moreover, by plugging (8) into (10), we find that δ is the solution of $\dot{\delta} = f(\delta)$. □

References

1. O. I. Aven, L. B. Boguslavsky, and Y. A. Kogan. Some results on distribution-free analysis of paging algorithms. *IEEE Transactions on Computers*, 25(7):737–745, July 1976.
2. O. I. Aven, E. G. Coffman, Jr., and Y. A. Kogan. *Stochastic Analysis of Computer Storage*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.
3. O. Babaoglu and D. Ferrari. Two-level replacement decisions in paging stores. *IEEE Trans. Comput.*, 32(12):1151–1159, Dec. 1983.
4. M. Benaïm and J. Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 65(11-12):823–838, 2008.
5. D. S. Berger, P. Gland, S. Singla, and F. Ciucu. Exact analysis of TTL cache networks. *Performance Evaluation*, 79:2–23, 2014.
6. J.-Y. L. Boudec. The stationary behaviour of fluid limits of reversible processes is concentrated on stationary points. *arXiv:1009.5021*, 2010.
7. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM'99*, volume 1, pages 126–134. IEEE, 1999.
8. H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: modeling, design and experimental results. *IEEE J.Sel. A. Commun.*, 20(7):1305–1314, 2002.
9. A. Dan and D. Towsley. An approximate analysis of the LRU and FIFO buffer replacement schemes. *SIGMETRICS Perform. Eval. Rev.*, 18(1):143–152, Apr. 1990.
10. R. Fagin and T. G. Price. Efficient calculation of expected miss ratios in the independent reference model. *SIAM J. Comput.*, 7:288–296, 1978.
11. N. C. Fofack, P. Nain, G. Neglia, and D. Towsley. Analysis of TTL-based cache networks. In *Valuetools 2012*, pages 1–10. IEEE, 2012.
12. C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971.
13. C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for lru cache performance. In *Proceedings of the 24th International Teletraffic Congress*, ITC '12, pages 8:1–8, 2012.
14. M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy. Performance evaluation of the random replacement policy for networks of caches. *Performance Evaluation*, 72(0):16–36, 2014.
15. N. Gast and B. Van Houdt. Asymptotically Exact TTL-Approximations of the Cache Replacement Algorithms LRU(m) and h-LRU. preprint, https://hal.inria.fr/hal-01292269, Mar. 2016.
16. E. Gelenbe. A unified approach to the evaluation of a class of replacement algorithms. *IEEE Trans. Comput.*, 22(6):611–618, June 1973.
17. J. H. Hester and D. S. Hirschberg. Self-organizing linear search. *ACM Comput. Surv.*, 17(3):295–311, Sept. 1985.
18. R. Hirade and T. Osogami. Analysis of page replacement policies in the fluid limit. *Oper. Res.*, 58:971–984, July 2010.
19. P. Jelenkovic. Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *Ann. Appl. Probab.*, 9(2):430–464, May 1999.
20. S. Jiang and X. Zhang. LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance. *SIGMETRICS Perform. Eval. Rev.*, 30(1):31–42, June 2002.
21. T. Johnson and D. Shasha. 2Q: A low overhead high performance buffer management replacement algorithm. In *VLDB '94*, pages 439–450, San Francisco, CA, USA, 1994.

22. J. Jung, A. W. Berger, and H. Balakrishnan. Modeling TTL-based internet caches. In *INFOCOM 2003*, volume 1, pages 417–426. IEEE, 2003.
23. W. F. King III. Analysis of demand paging algorithms. In *IFIP Congress (1)*, pages 485–490, 1971.
24. T. Kurtz. *Approximation of population processes*. Society for Industrial and Applied Mathematics, 1981.
25. N. Laoutaris, H. Che, and I. Stavrakakis. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.*, 63(7):609–634, July 2006.
26. E. Leonardi and G. L. Torrisi. Least recently used caches under the shot noise model. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2281–2289. IEEE, 2015.
27. A. W. Marshall and I. Olkin. *Inequalities: theory of majorization and its applications*. Academic Press, New York, 1979.
28. V. Martina, M. Garetto, and E. Leonardi. A unified approach to the performance analysis of caching systems. In *INFOCOM 2014*, pages 2040–2048, 2014.
29. B. V. H. Nicolas Gast. Transient and steady-state regime of a family of list-based cache replacement algorithms. In *Proceedings of ACM SIGMETRICS*. ACM, 2015.
30. E. J. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In *INFOCOM'10*, pages 1100–1108, Piscataway, NJ, USA, 2010. IEEE Press.
31. D. Starobinski and D. Tse. Probabilistic methods for web caching. *Perform. Eval.*, 46(2-3):125–137, Oct. 2001.
32. S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12, 2013.
33. N. Tsukada, R. Hirade, and N. Miyoshi. Fluid limit analysis of FIFO and RR caching for independent reference models. *Perform. Eval.*, 69(9):403–412, Sept. 2012.
34. J. van den Berg and A. Gandolfi. LRU is better than FIFO under the independent reference model. *Journal of Applied Probability*, 29(1):pp. 239–243, 1992.
35. J. van den Berg and D. F. Towsley. Properties of the miss ratio for a 2-level storage model with LRU or FIFO replacement strategy and independent references. *IEEE Trans. Computers*, 42(4):508–512, 1993.
36. S. Vanichpun and A. M. Makowski. Comparing strength of locality of reference – popularity, majorization, and some folk theorems. In *INFOCOM*, 2004.
37. S. Vanichpun and A. M. Makowski. The output of a cache under the independent reference model: Where did the locality of reference go? *SIGMETRICS Perform. Eval. Rev.*, 32(1):295–306, June 2004.
38. R. D. Yates. A framework for uplink power control in cellular radio systems. *Selected Areas in Communications, IEEE Journal on*, 13(7):1341–1347, 1995.
39. M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of YouTube network traffic at a campus network - measurements, models, and implications. *Comput. Netw.*, 53(4):501–514, Mar. 2009.